(2)

AD-A237 173

REPORT OF THE AMC SOFTWARE TASK FORCE

DTIC
S ELECTE JUN 25 1991 D
D

As Army systems come to increase their reliance on computers, software complexity increases and its criticality grows. Although progress has been made in recent years, the Army still does not have a cogent software acquisition and support policy nor does it have a strategy for insertion of new technology in a controlled, purposeful manner. This report builds on numerous previous studies to identify issues which confront the Army, documents underlying problems which the Army must face through this and into the next century, and identifies actions which must be taken to correct the problems.

James A. Hess, Chairman
Peter Fonash
Chris Neubert
Rcnald Keenan
Gerry Smith

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

BEST AVAILABLE COPY

081 ✓ 91-03133

91 6 21 081

28 February 1989

EXECUTIVE SUMMARY

Computer software is a necessary element in Army weapon systems but that software is contributing an inordinate amount of cost and risk to system acquisition. This report identifies the problems which the Army is facing and will continue to face unless corrective actions are taken. The report builds on and is consistent with a number of previous studies, investigations, and reports. It identifies recommended actions and calls on the Army to establish an orderly, task-oriented assault on the problems.

The report suggests there are three primary challenges to the Army: reduce the growth in the cost of software the Army is acquiring, bring the process to develop and maintain software under control, and maintain and extend the technological superiority of Army weapon systems through a focused investment in software. A taxonomy of software problems is presented which groups today's software problems into five areas: capability of people, absence of a clear and cogent software policy, lack of process controls, an absence of adequate procedures, and failure to capitalize on and plan for technology.

The report recommends an integrated course of action to try to gain control of the software in the Army's software systems. The recommendations call for the Army to: develop a strategy for software acquisition, organize to better manage the acquisition and support of software intensive systems, establish controls to guide programs and manage system acquisitions, and allocate sufficient resources for mission critical software. An implementation strategy is suggested which provides for intensive, dedicated management until significant progress is demonstrated.

Accesion For
NTIS CRA&I ☑
DTIC TAB □
Unannounced □
Justification ....
By ....
Distribution /
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 5

TABLE OF CONTENTS

SECTION I - Introduction

SECTION II - Problems

SECTION III - Recommendations

SECTION IV - Implementation

SECTION I
Introduction

"Many previous studies have provided an abundance of valid conclusions and detailed recommendations. Most remain unimplemented. If the military software problem is real, it is not perceived as urgent. We do not attempt to prove that it is, we do recommend how to attack it if one wants to".

Report of the Defense Science Board
Task Force on Military Software
July 1, 1987

1.1 BACKGROUND

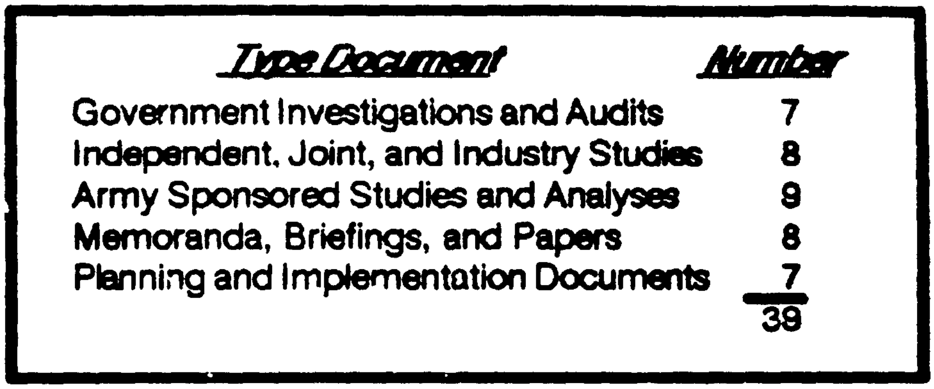
Computer software used in Army weapons systems is a critical element in today's battlefield. It is the computer with its requisite software which serves as a force multiplier. In far too many instances, however, computer software problems appear to have caused systems to be delivered late, over budget, and with severe problems which appeared during testing or in the field. The challenge to the Army is straightforward. It must ensure mission critical software is planned, designed, developed, acquired, integrated, tested, fielded, and supported so the Army can meet its battlefield objectives. For this to happen the software must meet quality and performance requirements, be available on a timely basis, and impose an acceptable life cycle cost burden on the Army. Some software initiatives have been taken to these ends, but much more remains to be done.

Although progress has been made in recent years, the Army still does not have a cogent software acquisition and support policy nor does it have a strategy for insertion of new technology in a controlled, purposeful manner. Over the past several years, aspects of the Army's software engineering process have been documented in: Army Post Deployment Software Support concept plans, Joint Logistics Commanders sponsored government/industry workshops and reports, Army and Defense Science Board studies, various Army Audit Agency and Inspector General reviews, and a variety of DoD/service sponsored studies of software management. Many of the issues are clear. This Task Force has been established tc address them.

The AMC Software Task Force was chartered to document the software problems which will confront the Army through the remainder of the century and identify initiatives which must be taken to correct them. It has identified mission critical software problems, categorized the problems into action areas, proposed recommended actions, and prepared a plan to bring software under control. The Task Force built on previous studies to the maximum extent possible; used a broad perspective tc address all aspects of the software problem; and developed a focused, orderly, task-oriented plan to address Army software problems.

1.2 PLOWING OLD GROUND

As noted in the 1987 Defense Science Board Task Force Report on Military Software, there has been a plethora of previous studies, investigations, and reports documenting the problems with battlefield systems and offering solutions to the U.S. military software problem. This Task Force did not attempt to reinvent all of the work that has gone before. Much of it was very good. Rather, the approach taken was to review the earlier findings, conduct further investigation where necessary, and assemble a coherent, comprehensive set of issues and pro lems. Approximately 40 documents were identified to serve as a basis for this effort. Figure 1.1 shows the number of documents which fell into each of five categories. The specific references are listed in Section V of this report.



Type Document	*Number*
Government Investigations and Audits	7
Independent, Joint, and Industry Studies	8
Army Sponsored Studies and Analyses	9
Memoranda, Briefings, and Papers	8
Planning and Implementation Documents	7
	39

Figure 1.1 – Source Documents

1.3 TASK FORCE METHODOLOGY

The Task Force went through a four step process in order to formulate its recommendations and propose an approach to implement the proposals. Issues were identified, the issues were analyzed to determine underlying problems, solutions were proposed, and a management approach for implementation was suggested. Each of these steps is described in more detail below. Several categories for issues, problems, and recommendations were defined. There was, however, no 1-to-1 mapping of issues-to-problems and problems-to-recommendations.

The issues and their solutions transcend simple categories. For example, training and education issues may stand alone, but their implications ripple into areas such as senior leadership awareness of software issues and the ability to make informed decisions on policy, funding, and planning. The best way of representing the interrelationships between these issues is with a network diagram as shown in Figure 1.2. In order to provide visibility into these relationships during the study, a database was constructed which establishes and tracks the linkages. The database contains the full text of the recommendations and shows the problems and issues they were intended to solve. It is recommended that this database be used as one of the management tools to assess and track the effectiveness of the implementation of this study. The steps the Task Force followed are outlined below.

1.3.1 Issue Identification.

Existing problems in the development and support of mission critical computer software were identified and validated. Maximum use was made of existing studies/surveys.

a. Previously completed studies, plans, and reports were collected and the issues identified in them were cataloged. These studies included PDSS/LCSE concept plans, JLC Orlando I/II reports, Army Ada introduction plans, Army/Defense Science Board software studies, GAO/AAA/DAIG software reviews, and others as shown in Section V.

b. Issue information was updated with data from ongoing management and software cost identification efforts.

c. Issues were categorized into the five areas of concern identified in the Task Force's Charter (people, policy, process, procedures, and planning), recent or ongoing actions to resolve them were identified, and the issues were used to define underlying problem areas.

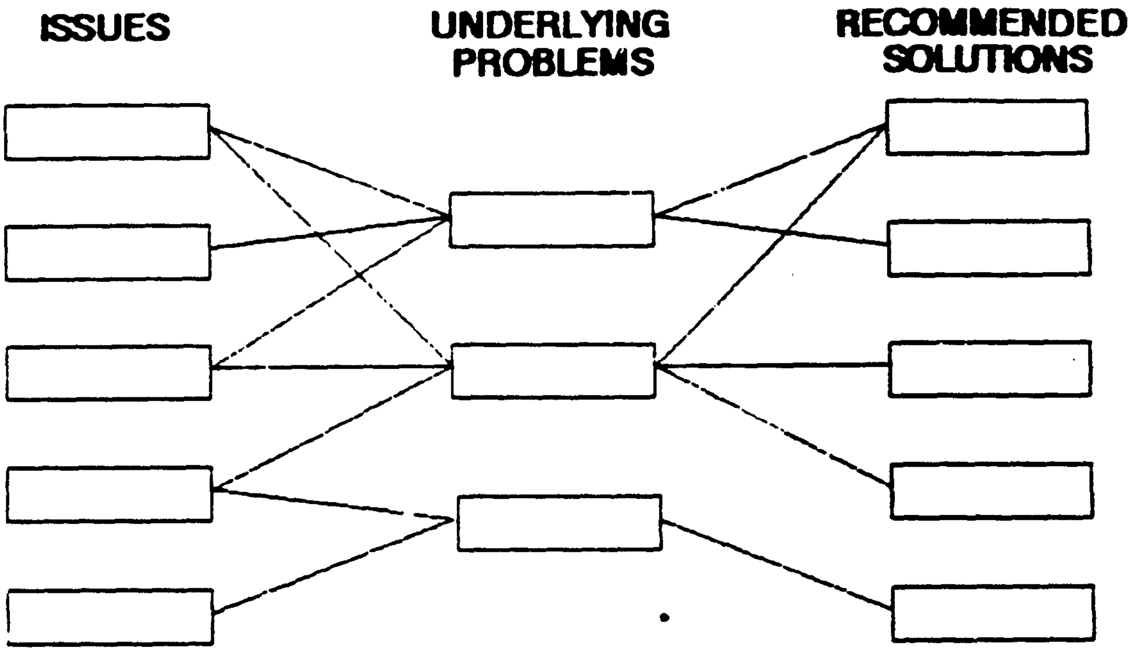


Figure 1.2 - Software Task Force Products

1.3.2 Problem Analysis.

A comprehensive analysis of the Army's software problems was prepared, additional information needed to completely understand the problems was identified and collected, and obstacles to problem resolution were defined.

a. On-site reviews were conducted to collect additional information needed for problem definition.

b. Meetings were held with industry consultants to compare interim findings of Task Force with earlier findings of the Software Engineering Institute, the Army Science Board, and the Defense Science Board.

c. The set of problems was reviewed with the study sponsor.

d. The problem definition was validated with a Review Group of Army software experts to ensure complete coverage, relevance of problems identified, and determination of any initiatives underway which were missed by the Task Force.

1.3.3 Proposed Solutions.

An integrated course of action to: develop a software strategy, establish policy and process, integrate action across the Army, and enforce software policy and procedure was developed. The recommendations were separated into work packages and roadblocks which prevented an earlier implementation of the recommendations were identified.

a. Recommendations were prepared in four general areas: strategy, organization, controls, and resources.

b. The problems and proposed recommendations were reviewed with the study sponsor, internal Army experts, and selected industry consultants.

1.3.4 Implementation Management.

A top level management plan was prepared. It will provide for the resolution of problems identified in the study and better management of Army mission critical software.

a. An implementation action plan was prepared which identifies each proposed action and lays out a time frame for execution based on priority.

b. Obstacles which prevented earlier resolution of the problems were identified: statutes, regulations, organizational impediments, inertia, funding, personnel capability, lack of understanding, or technology.

c. A transition plan was prepared which provides a management structure and resources to ensure follow-on efforts are executed.

1.4 ISSUES

The software issues, as identified by review of previously conducted software studies and through a variety of interviews and discussions, were grouped into five general categories. Although categories were used to help clarify and classify the issues, some issues were difficult to classify because they had impact in multiple areas. The list presented below is the final list that was used to derive the underlying problem areas:

1.4.1 PEOPLE - Software capability of Army people

Recruitment and retention of software experts
Inadequate MCDS SW Knowledgeable Workforce
Failure to maintain critical mass of SW professionals
Army has not implemented new Computer Engineer series

Resource constraints preventing use of interns/coops
Available People Misapplied

Lack of career program for software engineers

No Development Program for Civilian SW Professionals
No career path for Military SW managers
Lack of Installation Level SW Journeyman Programs
Gov't Personnel Motivated Not to Tailor Acquisitions

Maintaining state-of-the-practice software skills

No Education of PM/Contractor on Benefits of Ada
In-House Skills don't Match Contractor Skills
Difficult to get new technology into SW centers
Lack of MCDS SW Application Experience
No Functional/Project SW Quality Training Program
Lack of Interoperability Expertise
Area Support Analysts can't be jack-of-all-trades

Software survival skills for managers

No Software Survival Program for GO/SES Managers
No Mid-Level Software Awareness Program

Outreach to personnel affected by software

No Program to Inform Congress of Army SW Initiatives
No Education of PPBS Players of Importance of SW
No Software Orientation for HW people
Responsibility for SW Support Analyst Training Unclear
Responsibility for Field Operator Training Unclear
Revised Operator Instructions not Available w/ new SW

1.4.2 POLICY - Existence of a clear, cogent, effective policy

Lack of common framework for MCCR policy

No effective Army proponent for MCCR
No MCCR Planning Framework
Mo MCCR Guide for PMs
Responsibility for Interface Problems Undefined
Responsibility for Interoperability Testing Unclear
Responsibility for SW Support Analyst Unclear

Plethora of overlapping, conflicting guidance

Army Policy/Regulations not Current with DOD Guidance
IV&V Role of LCSE Centers, QA, and TECOM Confusing
Confusion of Performance DT and IV&V

Policy out of tune with emerging technology

Army does not Employ any SW Risk Management Approach
Firmware given blanket treatment as software
Difficult to Apply DOD-STD-2168 to Ongoing Work
DoD Data Rights Requirements do not motivate Reuse

Policy undefined or not clear

No Requirement for Use of Environment
Inadequate DOD-STD-2167/2168/1467/1815 Implementation
No Approach Defined for Support of NDI
No Clear Interoperability Test Policies
No clear definition of who pays for interoperability
No Higher Level Review to Insure proper SW $ Support
Army has no strategy for SW support in wartime

Weak policy enforcement, compliance measurement

PM/End Item Manager Funding of MCDS Short of Requirement
Poor Management Control to Ensure Good SW Transitioned

  - Poor Management Control of Transitioned Documentation
- **No feedback, little learning from past mistakes**
  - No process to document lessons learned on systems
  - No strategy to manage SW program and report progress
  - No ongoing review of LCSE functions and operations

1.4.3 PROCESS - Software planning, budgeting, and cost control

- **No effective computer resource management**
  - Status of CRMPs for Army's 200 MCDS Undefined
  - CRMPs not Prepared Early Enough to have Impact
  - CRMPs Inadequately Prepared
  - No data base exists which identifies computer resources
  - CRMPs not Reviewed by CRWG
  - No use of CRMPs to Control Acquisition Plan Approval
  - CRMPs not Submitted to AMC for Approval
- **Vague, ill-defined, or undefinable requirements**
  - Poor Feedback between Requirements and Specifications
  - Insufficient time/resources to Develop SW Specs
- **Little realism in cost estimates for software**
  - No Accepted SW Resource Estimators
  - Management Document Requirements Waived to save Time/$
  - Poor Mechanism to Evaluate MCDS Change on Interoperability
  - Poor Mechanism to Evaluation Interoperability Change on MCDS
- **Ineffective tradeoff of requirements versus cost**
  - Inadequate Time/Resources for SW Development
- **Poor understanding of true software support cost**
  - No Identification/Justification of Resources in CRMPs
- **Correction, enhancement, and modification control**
  - MCDS Software Changes not Timely
  - SW Changes required to Fit into PIP Process
- **Budget process unresponsive to needs**
  - No Methodology to Prioritize SW Support Workload
  - Difficult to address multi-year funding of SW Support
  - Resource Requirements Inadequate to Support SW maintenance
  - Inadequate Software Technology Funding
- **Duplication of software functions/facilities**
  - No Documented Plan (5-yr) for LCSE Center Improvement
  - Support for Training Device SW may require new LCSE
  - Duplication of resources at co-located LCSE Centers

1.4.4 PROCEDURES - Control of software acquisition/support

- **No overall management and control of software**
  - HQ,AMC has abdicated its MCDS software management role
  - Perceived Overlap of SW Effort by Functional Elements
  - No Differentiation of Management for Types of SW
  - Management process aimed at eliminating judgment
  - No Master Plan for SW Commonality across MCDS
  - Poor Profit Margins for Army Custom SW
  - SW Warranties not used for Stabilized Common SW
  - Modification of NDI SW not Addressed
  - Materiel Release program oriented to HW not SW
  - SW contracts not tied to contractor capability measures
- **Little guidance for selection of software environments**

- No Standard SW Development Environment
- No Standard SW Support Environment
- No Standard SW Test Environment

Lack of definition/integration of software tools

- Proliferation of Test Tools
- Inadequate Simulators/Test Bed Standardization
- Lack of Standardized SW CM Tools
- Software Tool usage still in infancy

Lack of management of development and methodologies

- Lack of Usable SW Design Documentation
- DOD-STD-2167A still emphasizes waterfall
- Each SW Procurement Package Must "Start-from-Scratch"
- No Definition of Standard SW Documentation for NDI
- No Standard way to measure goodness of designs
- No SW Standardized Transition Plans
- No Standard Approach to Evaluate Non-Ada MCDS Design
- No SW Coding Standards for Non-Ada MCDS
- No SW Testing Standards for Non-Ada MCDS
- Inadequate SW Documentation
- Ineffective monitoring of Design Process

Transition to the use of Ada

- Poor Support for Ada Introduction
- No Standardization/Control of Ada SW Requirements
- No Master Plan to Upgrade Existing SW/Retrofit to Ada
- No Up-to-Date Ada Introduction/Management Plan

Interoperability of systems, HW/SW integration

- MCDS Interoperability Requirements Unclear
- Historical lack of standardization of MCDS Data Links
- Inadequate Test/Interface Criteria for SW Modules
- Lack of Common Embedded Processors
- Lack of Common Test Equipment
- Lack of Common Program Media
- Lack of Common Media Replication Equipment
- Lack of Common Support Computers/Environment
- Lack of Common Tape Loading Units
- Lack of Common Disk Units
- Lack of Common Communications Interface Processor

Enforcement of configuration control

- Approved MCDS Software Baseline not Adhered to
- No Standardization of SW CM Planning
- Inadequate tracing of Detailed Design from Specs
- Inadequate Local Auditing Procedures
- Lack of Standardized CM System/Procedures
- Contractor's non-deliverable SW uncontrolled
- No CM Status Accounting Requirements
- Inadequate SW Configuration/Change Control
- Lack of SW Configuration Control Boards
- Inadequate SW Configuration Audits/Reviews
- Lack of Standardized CM Documentation
- No plan to transition CM data base from contractors

Software Quality Assurance

- SQA standards not consistently invoked
- Inconsistent approach to SQA in LCSEs

Effectiveness of testing and IV&V

- No Design Techniques/Methodology for Test Tool ID

- MCDS Testing not Centrally Coordinated
- Lack of Simulators and Stimulators during Development
- Inadequate Review of Test Procedures/Results
- Inadequate Comprehensive/Disciplined Test Practices
- Failure to Define Quantitative Acceptance Tests
- Inadequate Resources/Procedures for Test Integration
- Poor Coordination of System/Inter-System Test Programs
- Deferment of Unit Level Testing Until Systems Test
- Interoperability Test Concepts Unclear
- Interoperability Test Plans/Procedures Unclear
- No Interoperability Test Bed for Development/Support
- Lack of Interoperability Testing during Development

Software replication, distribution, and control

- Unworkable SW Replication/Duplication/Installation
- Army Supply System Unresponsive to SW Distribution
- Untimely Field Replacement of Failed/Superseded SW
- Untimely Distribution of New SW Documentation/Training
- No provision to electronically transmit changes/updates
- Inefficient use of Personnel to Support SW Distribution

Integration of Life Cycle Software Centers

Closure of feedback loop with field

- AMC Support to Field seen as Unresponsive
- No Standards for Processing Field Users Trouble Reports
- Uneven Technical Support to Field from LCSEs
- No Single-face-to-field for SW Problems

1.4.5 PLANNING - Support for research and technology insertion

Overall plan for Army SW Technology Efforts

- Strategy not built on leveraging off private sector
- STARS Program not Apropos to Army Needs
- No advocate for software technology in Army

Insertion of new technology into systems

- Army State-of-Practice far Behind State-of-Art
- No Support for Technology Insertion
- Lack of Army-wide Software Technology Center

Definition of integrated software environment

- Inadequate Strategy to Use Commercial SW Tools
- No Definition of Minimum Toolsets
- No standards for interchange of data from tools exists
- Lack of Facilities for Rapid Prototyping

Ada inefficiency, lack of run time environment

- Ada Benefits/Problems/Issues Undefined
- No plan to address Ada/Non-Ada Integration
- No Performance Tests for Ada Compilers

No automated process for requirements definition

- Loss of Traceability from Specifications to Design

Software component definition, reusable software

- No Plan to Reuse SW in Army's MCDS
- Lack of progress in establishing basis for reuse
- Lack of OTS/COTS Ada-based SW to serve as reuse base
- No Program to Identify Common SW Modules
- No Plan to Develop OS/Application/support SW Library
- No Methodology to Manage Reusable SW Components
- No Methodology for Configuration Control of Modules

Lack of Reusable SW Modules
Lack of Rehostable SW
Lack of Retargetable SW
Lack of Common SW
No Incentives for Contractors to develop Reusable SW

High reliability/integrity software development

No guidelines for test driver certification

Use of distributed systems and parallel processing

No guidelines for modeling and simulation

SECTION II
Problems

"As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem and now that we have gigantic computers, programming has become an equally gigantic problem. In this sense the electronic industry has not solved a single problem, it has only created them - it has created the problem of using its product".

"The Humble Programmer"
E. W. Dijkstra
Turing Award Lecture, 1972

2.1 CHALLENGE TO THE ARMY.

Computer software plays a critical role in Army mission critical defense systems. DoD Congressional testimony has described software as follows:

> "Software is the human intelligence that is programmed into our systems. It allows advanced sensors to discriminate and track, navigation systems to follow prescribed routes, guidance systems to control trajectories, and communications systems to properly route thousands of messages. Software keeps track of the status of our forces, maintains intelligence information on enemy forces, and aids our commanders in deciding on target actions."

As the complexity of Army defense systems increases, mission critical software seems to be getting out of control. Software is the least understood and has the highest perceived risk of any aspect of systems development. It contributes an ever increasing share of acquisition risk and its aggregate cost grows exponentially. However, software offers the only hope for meeting today's requirements for battlefield command and control, and weapons system operation.

The capability and technology of Army software will determine who dominates the battlefield of the 1990's and beyond. It is the embedded computer with its requisite software which permits battlefield commanders to operate within the enemy's decision cycle. Software allows the Army to meet rapidly changing threats, incorporate sophisticated controls, and assist the commander in the conduct of his mission. The challenge to the Army is to effectively manage software and its technology so Army needs are met at minimum life cycle cost. The Army must bring the cost of its software under control, guarantee high quality in its software products, and maintain technological superiority of its software driven systems.

2.1.1 Cost of Software.

The amount of money spent on computer software has been growing at an exponential rate. It has been estimated that the annual cost of computer software and services for the Department of Defense will increase from $11.4 billion in FY-85 to $36 billion in FY-95. If current trends continue, computer software costs could exceed the entire defense budget by the year 2015. Whether this prediction is to be believed or not is immaterial. There are numerous dire predictions of software cost and complexity growth. Many of these predictions point to the high level of costs required to maintain software systems after deployment. Currently, between 50 and 75 percent of the software cost for Army weapon systems is expended for post deployment support, commonly known as software maintenance.

The Army first approved its concept for Post Deployment Software Support (PDSS) in 1978. Each year since then, resource requirements for software support have far exceeded the amount funded. Yet no impact of this lack of funding was evident. In some cases the requirements were overstated because they were based on inadequate cost prediction methods. This overstatement continues to hurt the credibility of the Army software community. In other cases, inadequate funding led to the deferral of important software corrections and enhancements. The most insidious effect of this funding profile, however, has been the deferral of software engineering considerations during system development. Poorly designed and inadequately documented software has been delivered to and accepted by the Army. These deficiencies have increased the level of effort needed to sustain the systems until the software could be reverse engineered and matured. In some cases the initial cost of sustaining software is more than double the cost reached at maturity. Each year the consequences of these deferments have been compounded. The Army is now faced with a large software support liability. In FY-89, the Army must either find additional funds to sustain existing fielded systems or drop software support for some of those systems. The situation will only get worse as additional software intensive systems continue to be fielded.

2.1.2 Product and Process Control.

As Army systems come to increase their reliance on embedded computers, software complexity increases and its critical impact grows. Software, in fact, is necessary to implement all of the

Army's current Command and Control initiatives. Thus, the responsiveness and reliability of our defense systems is software dependent. The challenge to the Army is to ensure embedded software is planned, developed, acquired, integrated, tested, fielded, and supported so the Army can meet its battlefield objectives. For this to happen the software must be available on a timely basis and meet quality and performance requirements.

A direct result of the order of magnitude growth per decade in the requirements for software has been a growing shortage in the number of qualified software engineers. The need for software engineers has grown at a rate of approximately 12% per year during the 80's. This has resulted in a shortage of qualified people since the increase in new practitioners has been only 4-6% per year. For a variety of reasons discussed later, the situation within the Army is even worse. It is critical for the Army to find a means of increasing the productivity of its software development and support activities and to more efficiently use the people which are available.

Some believe significant increases in software productivity can be achieved solely through the introduction of the latest technology. This, however, is a misconception. An effective approach for software productivity involves much more than the introduction of "modern programming practices." For example, experience with a computer language or in an application area, the use of powerful software tools or modern programming practices, and the effect of introducing stringent design goals for reliability provide only a minimal productivity improvement when considered individually. Productivity enhancing opportunities must be sought and management actions taken throughout the entire software life cycle if the software process and its products are to be brought under control.

2.1.3 Technological Advantage.

It can be argued that America's greatest military strength lies in the individual initiative and ingenuity of our Service people. The fruits of their initiative and ingenuity are bolstered by the technological superiority of our defense systems. This need for technological superiority makes computer technology especially critical. Because Army conventional forces are outnumbered in almost all areas on the battlefield, they rely on mission critical computer systems to serve as a force multiplier. Software based command and control systems help to provide the knowledge base which fosters initiative in battle plannirg. They allow a more effective deployment of Army troops. Embedded computers operating in real-time can and do dramatically increase the lethality of the weapons into which they are incorporated. The relative ease of adapting software provides the ability to introduce enhancements more quickly and at less cost than making hardware changes. Software can extend the useful life of a system by incorporating incremental improvements and quickly responding to changed threats. In some cases software has been used to overcome defects in original hardware design. Most important of all, however, is the advantage the U.S. holds in the area of software development. Because the American technological lead has been shrinking in other areas, it is essential to maintain and increase the ability to capitalize on software technology.

Fortunately, maintaining a software technology advantage does not require huge expenditures of Army research and development funds. The pervasive use of the microcomputer and the development of a mass market for software technology has spurred the development of the industrial base for software technology. The challenges to the Army, therefore, are not so much technical as they are managerial. A strategy to capitalize on and use advances made by industry is desperately needed. The Army cannot let its software advantage slip away; instead it needs to manage and effectively insert new technology to increase its advantage.

2.2 TAXONOMY OF PROBLEMS.

The Task Force felt the Army must abandon any search for a magic potion to bring about a marked reduction in the cost of software, improvement of product and product control, and maintenance of technological superiority. Such a tool or technique does not exist. In actuality, a variety of institutional, political, and socioeconomic problems must be addressed before the Army can improve its control and management of software. Instead, the Army must emphasize sound management practices and the selected use of new technologies as their worth is proven. Decisive and coordinated actions need to be taken to improve the software capability of Army people, establish a clear and cogent software policy, develop better controls and procedures to control software acquisition and support, and plan for capitalization and insertion of appropriate software technology into the way the Command does business. A unified productivity improvement program needs to be initiated. The formula for success is very simple:

```
PRODUCTIVITY == PEOPLE IMPROVEMENT and
                POLICY IMPROVEMENT and
                PROCESS IMPROVEMENT and
                PROCEDURE IMPROVEMENT and
                PLANNING IMPROVEMENT
```

Note that productivity is more than the sum of improvements made in five specific problem areas. It is their logical conjunction. Failure to se: gains in any one of the areas can obviate improvements made in the other four. For example, the clearest policy, the best procedures, and the most up-to-date tools can be undone if the workforce is untrained or unmotivated. A taxonomy was developed and used to classify the problems into the five areas essential for real productivity improvement and then according to twenty three observable effects. Figure 2.1 illustrates the taxonomy structure. The causes shown on the diagram correspond closely with the issues identified in Section I of this report. Eighty five problems were identified. Their taxonomy and a description of each is contained in Appendix A, but summary of each of the problem areas, their effects, and a pointer the examples in Appendix A is presented below.

2.2.1 People.

People efforts need to focus on building a new software cadre while addressing the existing work force both military and civilian. Action must be taken to establish software engineering

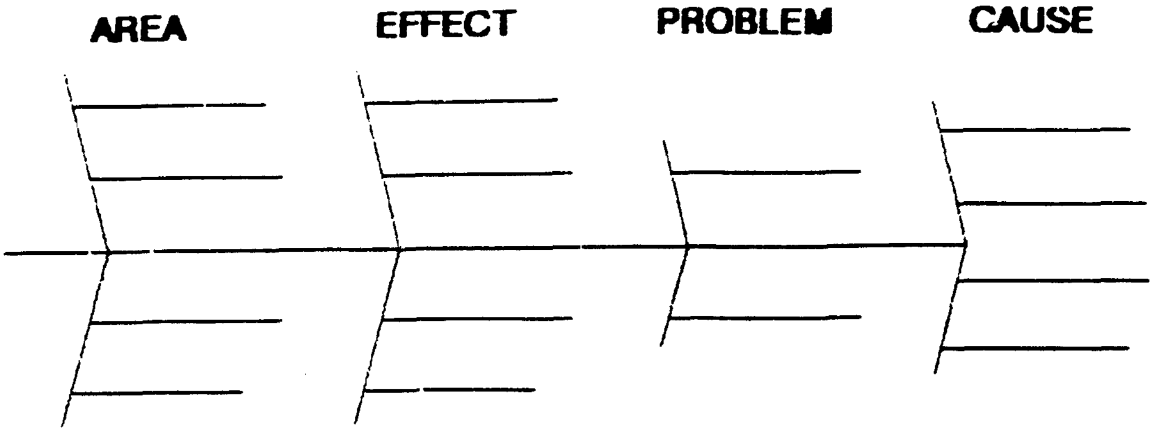


Figure 2.1 – Software Problem Taxonomy

as critically important and strongly supported throughout the Army. Programs to develop software interns[4], provide career paths for careerists, train managers with appropriate software management skills, and satisfy continuing education requirements must be pursued. The following specific problems ought to be attacked:

Little awareness of Software Criticality to the Army (A1-A5)

Difficulty in Recruiting Software Talent (A6-A10)

Mid Career loss of Software Talent (A11-A13)

Inadequate Software Training Program (A14-A18)

Inadequate Software Education Program (A19-A21)

2.2.2 Policy

Numerous regulations, pamphlets, letters, and guidebooks exist which are supposed to guide the project manager in the acquisition of computer hardware and software in the Information Mission Area. In general, however, Project Managers are not aware of what is required or where to go for help. The policy needs to be set in a common framework. Because the Army cannot afford to be locked out of the mainstream of software technology, provisions need to be made to keep the policy in tune with emerging technology and practice. The policy, however, must remain consistent over time and a mechanism must be put in place to enforce software policy, measure compliance, and develop lessons learned so we can learn from our mistakes. The solutions to problems in four specific areas need to be found:

Software treated as if it were monolithic (A22-A24)

Gaps and Inconsistencies in Software Policy (A25-A27)

No Evaluation of Software Policy Effectiveness (A28-A30)

Little Policy Evolution for Emergent Methods (A31-A33)

2.2.3 Process.

Processes for software planning, budgeting, and cost control are out of control. Controls need to be established to effectively manage the acquisition of computer resources and to provide realism in cost estimating. Key problem areas for resolution include:

System Requirements poorly defined and executed (A34-A37)

Ineffective Computer Resource Management (A38-A40)

Funding Inadequate for Software Support (A41-A45)

Responsibilities for Software Unclear (A46-A49)

Ineffective Software Management Process (A50-A53)

2.2.4 Procedures.

Procedures which impact Army systems have received inadequate consideration. Efforts must be mounted to address earlier and more fruitful interaction with the user, a cleaner handoff to contractors where appropriate, and methods to enhance the maintenance process done in-house. Procedural improvements in at least the following six areas ought to be pursued:

Inadequate Software Management Process (A54-A57)

Unconstrained Software Environment (A58-A61)

Lack of Enforceable Standards (A62-A65)

Need to account for Hardware/Software Differences (A66-A69)

Lack of coordination across matrix Organizations (A70-A71)

Untimely response to field software Problems (A72-A75)

2.2.5 Planning.

A strategy for the investigation and eventual insertion of new software technology must be developed. Candidate items ought to include methods for realistic software resourcing, methodologies to permit more effective maintenance of software once delivered, and a mechanism for testing and inserting new software tools as appropriate. Key deficiency areas include:

Army has no defined software technology strategy (A76-A79)

Failure to capitalize on software advances (A80-A83)

Technology insertion measures ineffective (A84-A85)

SECTION III
Recommendations

"Perhaps the most important principle on which the economy of a manufacture depends, is the division of labour amongst the persons who perform the work. The division of Labour suggests the contrivance of tools and machinery to execute its processes".

"On the Division of Labour"
Charles Babbage, 1832

3.1 TAKING THE INITIATIVE.

Army software initiatives have been the direct result of a variety of panels and groups which have examined the software problem for the DoD and the Army. In the mid-70's the DoD Software Management Steering Committee recognized the problems inherent in a wide diversity of programming languages and, as a result, initiated work on a language which was later named Ada. The Computer Resource Management Joint Policy Coordinating Group of the Joint Logistics Commanders sponsored the Monterey I & II workshops with industry which identified the need for common software standards and the Orlando I & II workshops which addressed life cycle software support issues. The Army conducted a study of the way it supported embedded software which led to the Post Deployment Software Support concept in the late 70's and the Army Science Board suggested the need to address software during the entire life cycle which led to the Life Cycle Software Engineering program. The primary facilitator and implementor of these initiatives within the Army has been AMC. These and other initiatives have enabled the Army to make real progress in

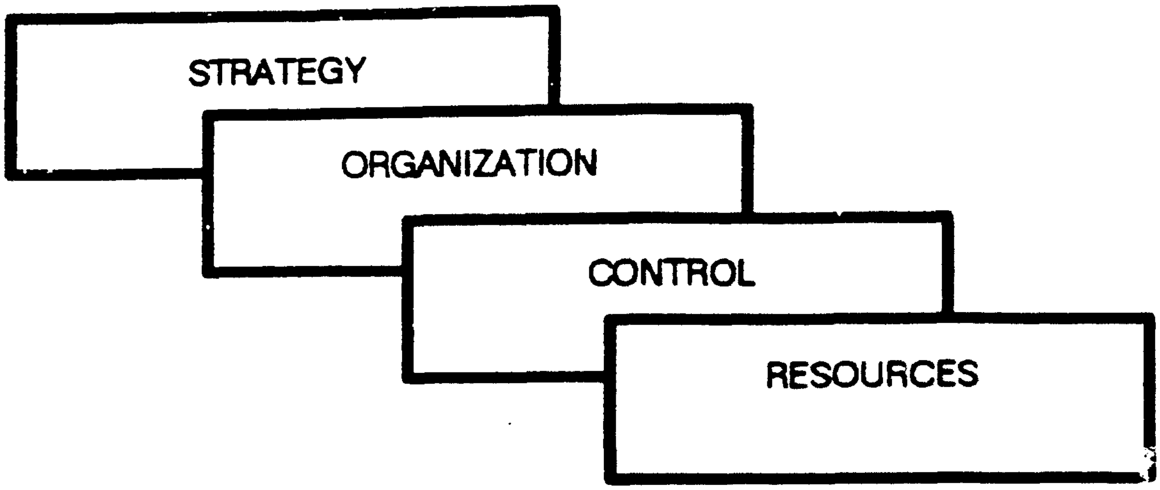


Figure 3.1 – Recommendation Areas

addressing problems in software productivity, management, and quality. Now, however, momentum has been lost. AMC Headquarters has abdicated its role in the management of Missi n Critical software. This study recommends an integrated course of action to try to gain control of the software in the Army's systems. As Figure 3.1 illustrates, the recommendations have been grouped into four large work packages. Briefly, the recommendations, which are listed in Appendix B, call for the Army to:

a. develop a strategy for MCCR Acquisition,

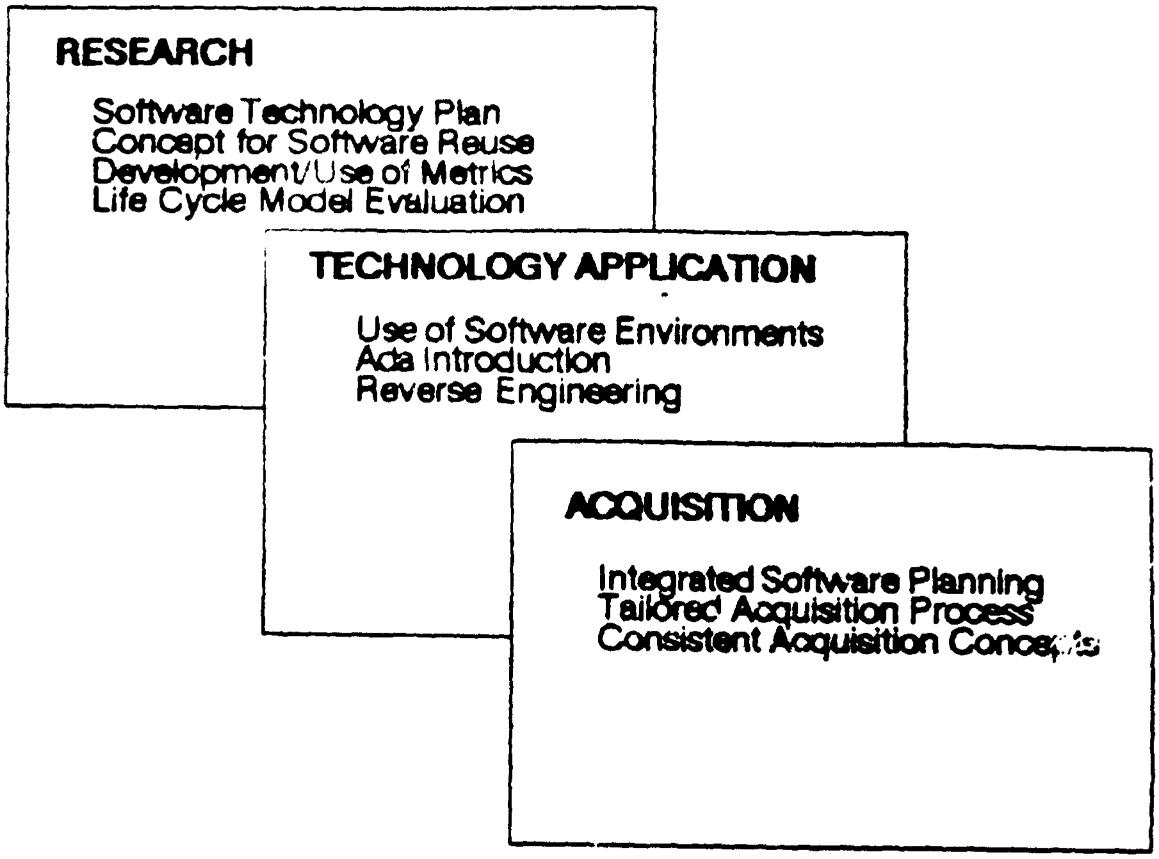
b. organize to manage the acquisition and support of software intensive systems,

c. establish controls to guide programs and manage system acquisitions, and

d. allocate sufficient resources for mission critical computer resources.

3.1.1 Develop a Strategy.

As shown in Figure 3.2, the recommendations for developing an



RESEARCH

Software Technology Plan
Concept for Software Reuse
Development/Use of Metrics
Life Cycle Model Evaluation

TECHNOLOGY APPLICATION

Use of Software Environments
Ada Introduction
Reverse Engineering

ACQUISITION

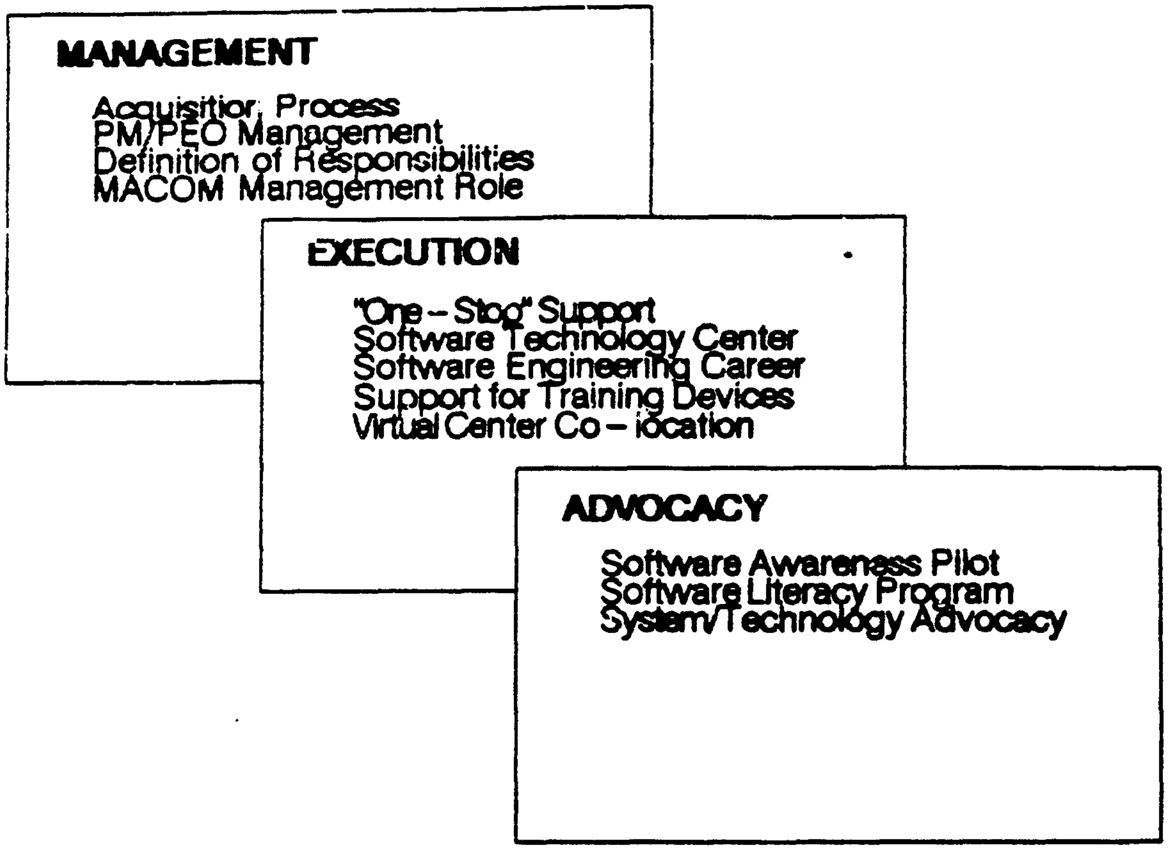
Integrated Software Planning
Tailored Acquisition Process
Consistent Acquisition Concepts

Figure 3.2 – Develop a Strategy for MCCR Acquisition

overall strategy for MCCR acquisitions are sub-divided into three smaller packages. The first calls for building a strategy to conduct research in the area of software engineering based on getting the maximum leverage off advances made by industry. The second part of the strategy should identify ways in which the Army can apply at least state-of-the-practice technologies to its systems. Finally, the Army must look at its acquisition process to provide for technology insertion.

3.1.2 Organize to Manage.

Figure 3.3 shows three aspects of the Army organizational structure which must be improved in order to better manage



MANAGEMENT

- Acquisition Process
- PM/PEO Management
- Definition of Responsibilities
- MACOM Management Role

EXECUTION

- "One – Stop" Support
- Software Technology Center
- Software Engineering Career
- Support for Training Devices
- Virtual Center Co – location

ADVOCACY

- Software Awareness Pilot
- Software Literacy Program
- System/Technology Advocacy

Figure 3.3 – Organize to Manage Software Intensive Systems

mission critical software. First of all, organizational missions and roles in the area of mission critical software need to be clarified and streamlined. Second, our executing commands and their software centers need tighter controls and better interfaces. Finally, and perhaps most critical, the Army needs to develop advocates who understand the critical role of software in today's Army and can effectively fight for the resources the Army must invest in this area.

3.1.3 Establish better Controls.

Figure 3.4 identifies the two main types of recommendations

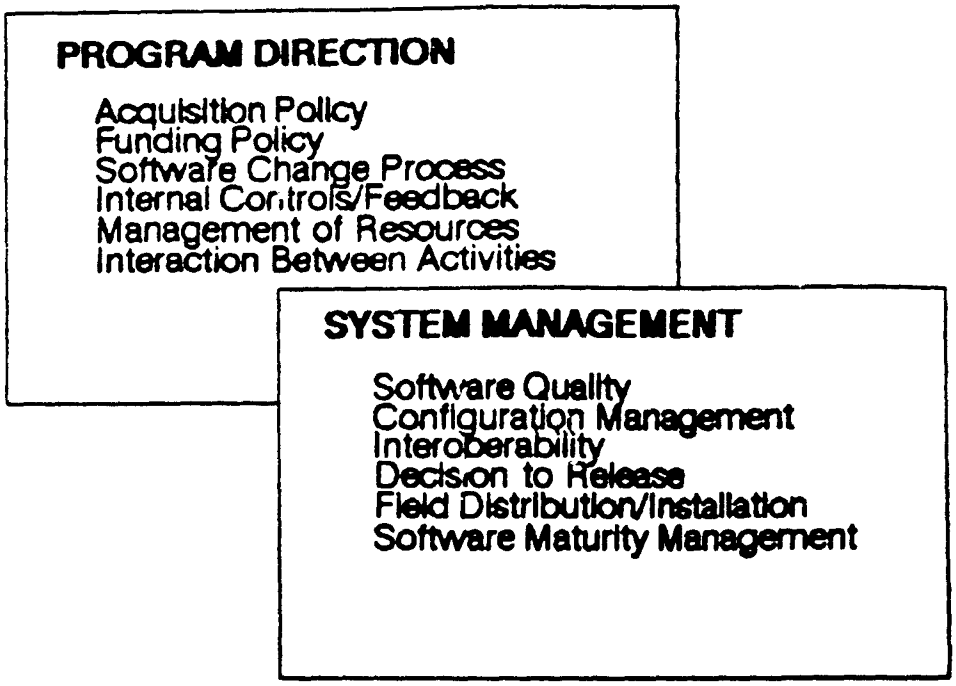
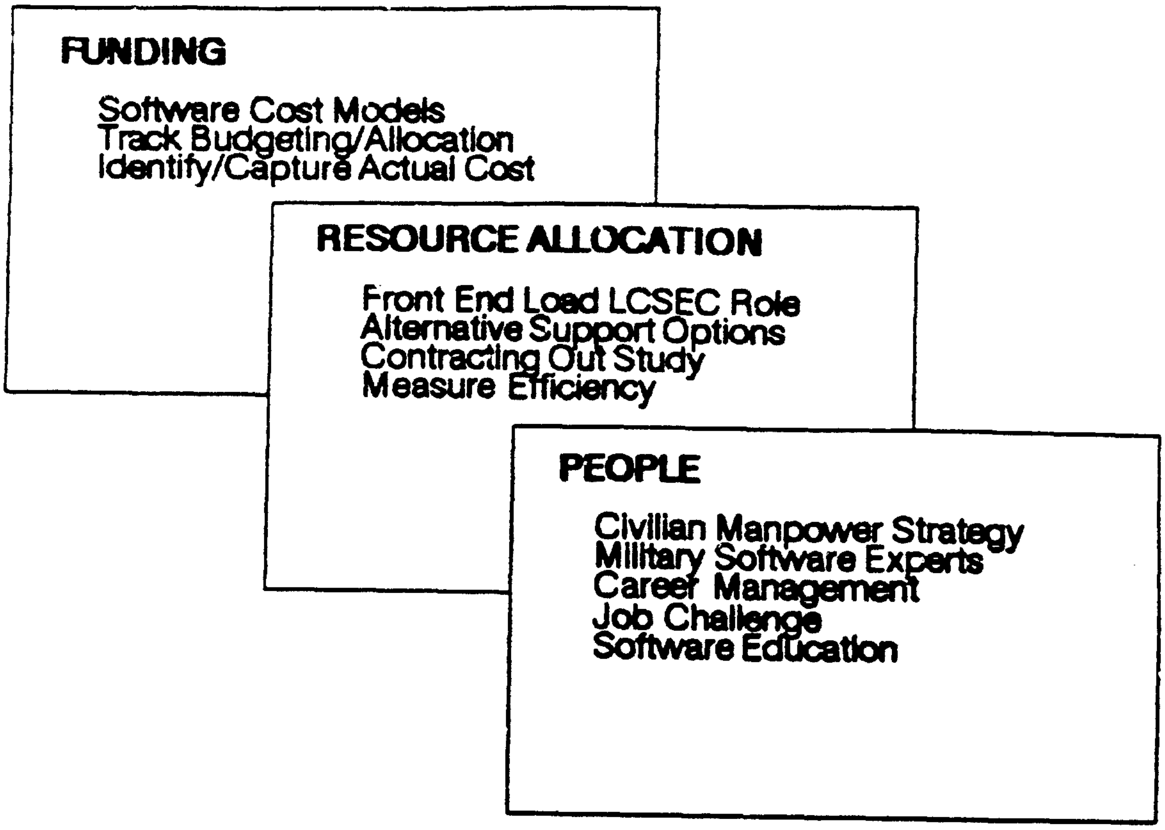


Figure 3.4 – Establish Controls for MCCR Program/Systems

concerned with the establishment of more effective controls for mission critical computer systems. The first group of recommendations is focused on those aspects of policy and enforcement necessary to more effectively provide direction to acquisition programs. The second group of recommendations is directed at the tasks which must be performed in certain key functional areas.

3.1.4 Resource Allocation.

There are three aspects of resource allocation which must be addressed. As shown in Figure 3.5, processes to identify and allocate sufficient funding for software activities need to be implemented, means to better allocate in-house people so they are not merely contract monitors must be found, and steps to better



FUNDING

Software Cost Models
Track Budgeting/Allocation
Identify/Capture Actual Cost

RESOURCE ALLOCATION

Front End Load LCSEC Role
Alternative Support Options
Contracting Out Study
Measure Efficiency

PEOPLE

Civilian Manpower Strategy
Military Software Experts
Career Management
Job Challenge
Software Education

Figure 2.5 – Allocate Sufficient Resources for MCCR

train and develop in-house people must be taken.

SECTION IV
Implementation

"If it were done, when 'tis done, then 'twere well it were done quickly".

"Macbeth"
William Shakespeare

4.1 NO MORE BAND AIDS.

There is no need to continue to study and restudy the software problems with mission critical computers. Over the past several years, problems with the Army's software engineering process have been documented in a variety of concept plans, workshops, reports and studies. This report has attempted to provide a unified view of the issues, problems, and their recommendations. The issues are clear. An integrated set of problems has been defined. Now resources must be found to address the recommendations in a focused effort. If the recommended actions are to be addressed, the Army can't depend on ad hoc, one-time projects. An orderly, task-oriented approach needs to be followed using dedicated in-house expertise, contract support as needed, and the cooperation and support of industry.

4.1.1 Planning Templates.

The Task Group analyzed and prioritized each of its recommendations. The results of that effort are contained in Appendix C to this report. Roadblocks which prevented earlier efforts in these areas have been identified. It is significant to note that only one recommendation, getting pay comparability with industry, requires statutory action. Many require regulatory or organizational changes, but an even larger number haven't been accomplished just because of inertia -- no one tried. There are also a large number of recommendations which need a significant amount of resources in order to be completed. Resources, in this context, could include: (1) funds, (2) technically qualified people, (3) reaching an understanding of new concepts, or (4) development of new technology. An approximate schedule showing when each of the recommendations could be addressed was also prepared. The scheduling reflects the Task Force's perception of the priority of the recommendations and implies a sequencing of tne tasks. It is not based upon a detailed analysis of resource requirements and availability, so therefore the times should be viewed only as a first cut approximation at this time.

4.1.2 Detailed Planning.

The first step in the implementation of the Task Force recommendations must be the development of a detailed plan for the execution of each task shown in Appendix B and the subsequent updating of the information in Appeudix C. The plan for each task ought to include the following:

a. Organization responsible for completion of the task and organizations which will provide support

b. Events which must be completed before a task is initiated, and tasks which are predecessors to it

c. Subtasks which comprise the task

d. Resources (time, money, people, and equipment) required to complete the task

e. Measures of success which will be used to determine when the task is completed

4.1.3 High Priority Tasks.

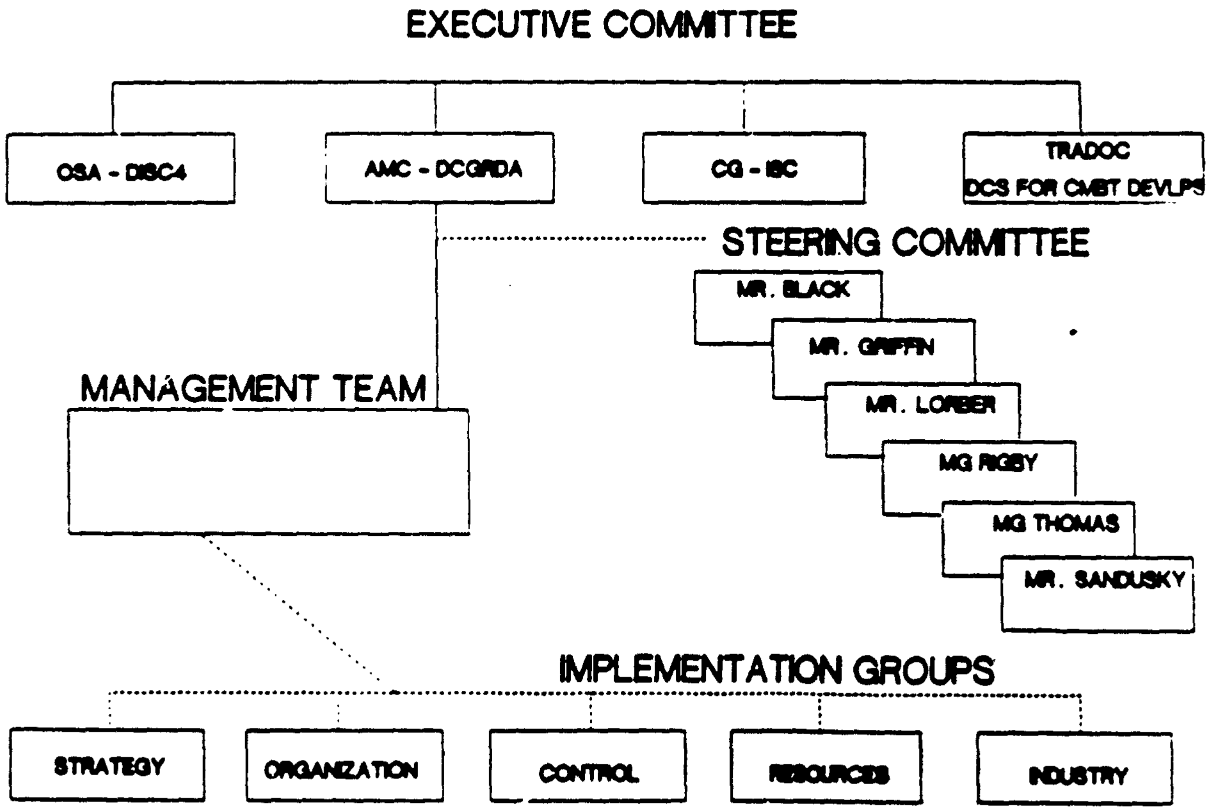
There are certain nigh priority tasks which will drive many of the actions taken in implementing this study. Efforts on them should be initiated in parallel with the detailed planning process. Those tasks are:

a.	SS-211A	Reorganize HQDA acquisition management
b.	SS-213	Clarify Organizational Responsibilities
c.	SS-132A	Define a streamlined CRMP process
d.	SS-315A	Develop a database with MCCR information
e.	SS-411A	Mandate the use of SECOMO for PDSS efforts
f.	SS-312A	Define BPRR/MAMP manpower process
g.	SS-323B	Accelerate funding of AIN
h.	SS-134C	Evaluate and use contractor performance info
i.	SS-321A	Provide for Total Quality Management
j.	SS-111A	Create software technology plan
k.	SS-222A	Reaffirm/staff Software Technology Center
l.	SS-422A	Define specific tasks to do in-house
m.	SS-121A	Establish standards for software environments
n.	SS-231A	Develop software story/lessons learned
o.	SS-223A	Establish software engineering program
p.	SS-223A	Prevent wholesale engineer reclassification
q.	SS-433C	Provide staff for software engineer subprogram

4.2 IMPLEMENTATION ORGANIZATION.

There is no one place in the Army where a sufficient number of qualified people can be found to implement the recommendations of this study in an integrated way. For that reason, a multi-organizational implementation group needs to be developed which can provide the focused and coordinated management which is needed. An implementation group structured as shown in Figure 4.1 is recommended. Each of the four major work packages discussed in Section III is essentially separable from the others and can be managed by a separate implementation group. The group responsible for resources should be run out of AMC, Headquarters. The others can be most effectively managed from within AMC's subordinate commands. Because of the need to gain the support of industry for many of the efforts, a fifth group is suggested to handle the interface with industry. Each group leader should be selected and identified by name and the group leaders should be responsible for proposing members of their team. It is important that the group members reflect a wide cross-section of the Army: representing all appropriate MACOMs and provide adequate functional diversity. The group leaders would be responsible for detailed implementation planning with their group and for recommending assignment of specific responsibilities for task execution. A management team is proposed at HQ, AMC to provide overall project management, identify resources required and allocated, track accomplishments, and keep all activities coordinated. A steering committee is recommended to provide integration across functional areas and an executive committee is proposed to provide inter-MACOM oversight. Although the completion of all the tasks identified in this report will extend over several years, it is recommended that the intensive management process suggested in Figure 4.1 have a one year sunset clause. After that time, the implementation accomplishments should be reviewed by the executive group. The executive group should decide whether to continue implementation with the intensive management process described herein or to switch to the normal management chain with periodic joint-MACOM progress reviews. In either case, the implementation will require close management attention and support for an extended period of time.

IMPLEMENTATION APPROACH



(Page intentionally left blank)

SECTION V
References

5.1 Government Investigations and Audits

BATTLEFIELD AUTOMATION: Field Artillery Data Systems Acquisition Problems and Budget Impacts, US GAO Briefing Report GAO/NSIAD-87-198BR, July 1987

Audit of Support for Tactical Software, U.S. Army Audit Agency, Report NE 78-209, 30 September 1987

Inspection of Army Program Management (Findings 36, 37, 38, & 39), US AMCIG, 6 April- 31 October 1987

Advisory Report on Tactical Software, U.S. Army Audit Agency, Report NE 88 A1, 30 June 1988

A Report to Congress on Central Design Activities, Assistant Secretary of Defense (Comptroller), August 4, 1988

COMPUTER LANGUAGE STANDARDIZATION: Status, Costs, and Issues Associated with Defense's Implementation of Ada, US GAO Draft Report GAO/IMTEC-89-9, 4 November 1988

Management of Software for Mission Critical Computer Resources, US DODIG Draft Report 7ID-077, 2 December 1988

5.2 Independent, Joint, and Industry Studies

1983 Summer Study on Acquiring Army Software, Army Science Board (ASB), 20 April 1984

Post Deployment Software Support (PDSS) for Mission-Critical Computer Systems, Final Report of the Joint Logistics Commanders' Workshop, June 1984

Summary of Products and Recommendations, Joint Logistics Commanders 4th Biennial Software Workshop - "Orlando II", June 1987

Report of the Defense Science Board Task Force on Military Software, Defense Science Board, 7 October 1987

Engineering Employment: Trends, Issues, and Recommendations, Institute of Electrical and Electronics Engineers USAB Issue Brief, June 1988

National Strategy Issues, Software Engineering Institute Workshop, 3 August 1988

Adapting Software Development Policies to Modern Technology, Air Force Studies Board, 19 August 1988

Software Problem Report, Software Engineering Institute Workshop, 31 October - 1 November 1988

5.3 Army Sponsored Studies and Analysis

U.S. Army In-Theater Post Deployment Software Support - Phase I Report, Teledyne Brown Engineering, 22 September 1982 to 22 March 1983

Life Cycle Software Support (LCSS) Mini Functional Area Assessment, Department of the Army, 25 February 1987

Software Management Working Group - Study Report, US Army Communications-Electronics Command, Report SD013-12, 12 June 1987

Army Implementation of DOD & Federal Standards: Automation Implementation Guidance, Planning Research Corporation (for US Army ISEC), 29 January 1988

AN/TTC-39 Circuit Switch, AN/TTC-39A Modal Control Circuit Switch, and AN/TYC-39 Message Switch, Task Force Report, US ARMY CECOM Center for Software Engineering, undated

Acquisition Improvement Review - Mission Critical Defense Systems Software, Army Materiel Command, 31 August 1988

A Strategy for Improving U.S. Army Software Quality, MITRE Corporation (for the US Army DISC4), October 1988

An Analysis of Existing Software Development Practices - Process Models and Methods, Task 1 Report, ISSI-Lockheed (for US Army CECOM), 3 January 1989

Firmware Maintenance/Changes to Support Communications and Electronic Systems, Draft Study Report, US ARMY CECOM Center for Software Engineering, 9 January 1989

5.4 Memoranda, Briefings, and Papers

Management of Mission Critical Computer Resources (MCCR), US ARMY DCSRDA Action Memorandum, 9 October 1986

Breaking Barriers, Joseph S. Greene, Ada Expo '87, Boston, MA, 8-10 December 1987

Technology for Assessment of Software Quality, US ARMY AMCCOM Briefing, 20 April 1988

An Automated Systems Engineering Approach to Army Battlefield Automated Systems (BAS) Management, Briefing to US ARMY AMCDRA, Teledyne Brown Engineering, 8 June 1988

Reducing Software Costs: A Corporate View, Briefing to US ARMY AMCDRA, US Army Communications-Electronics Command, 27 July 1988

Allocation of CECOM FY89 OMA P2 Funds for Post Deployment Software Support, US ARMY CECOM Center for Software Engineering, Ft. Monmouth, NJ, 26 September 1988

Contractor Software Capability Evaluation, Briefing to the Joint Logistics Commanders, Software Engineering Institute, 27-29 September 1988

Documenting Management of MCCR in Battlefield Automated Systems, Briefing to US AMCDRA, IIT Research Institute, 11 October 1988

5.5 **Planning Documents**

Software Technology for Adaptable, Reliable Systems (STARS): Program Strategy, US DOD, 15 March 1983

Life Cycle Software Support (LCSS) Implementation Plan, HQ Department of the Army, 13 Jan 1984

Life Cycle Software Management Study: Implementation Plan, US ARMY CECOM, 3 June 1987

Action Plan, Joint Logistics Commanders 4th Biennial Software Workshop - "Orlando II", March 1983

Managing Information - Long Range Plan: 1988-2008, US ARMY DISC4, April 1988

Ada Technology for Command & Control, Presentation by the AWIS Technology Council, 21 September 1988

Advanced Software Technology, Organization and Operation Plan, US ARMY CECOM Center for Software Engineering, undated

APPENDIX A

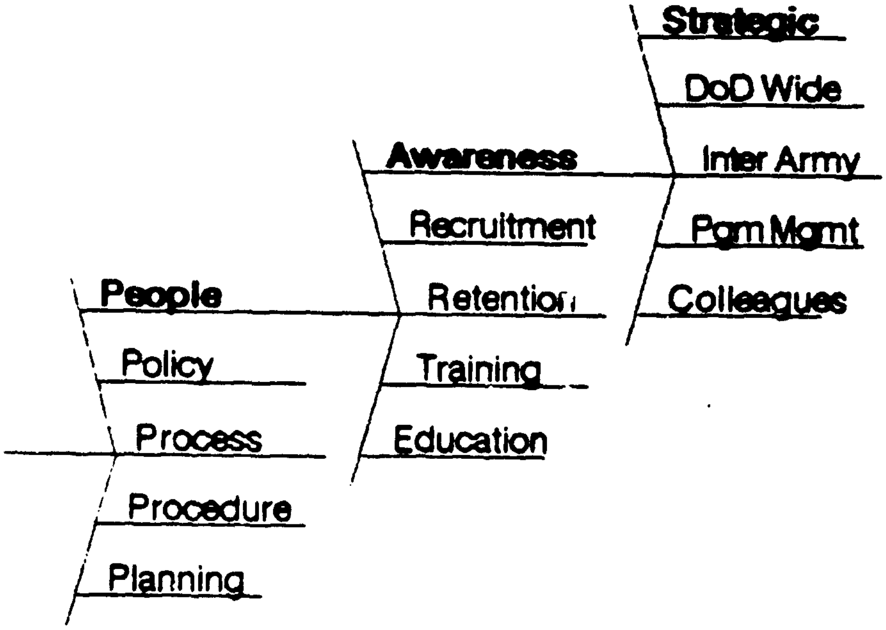
Software Problem Taxonomy

"Today we tend to go on for years, with tremendous investments to find that the system, which was not well understood to start with, does not work as anticipated. We build systems like the Wright brothers built airplanes – build the whole thing, push it off the cliff, let it crash, and start over again."

R. M. Graham
NATO Science Committee
conference on Software Engineering
October 7 – 11, 1968

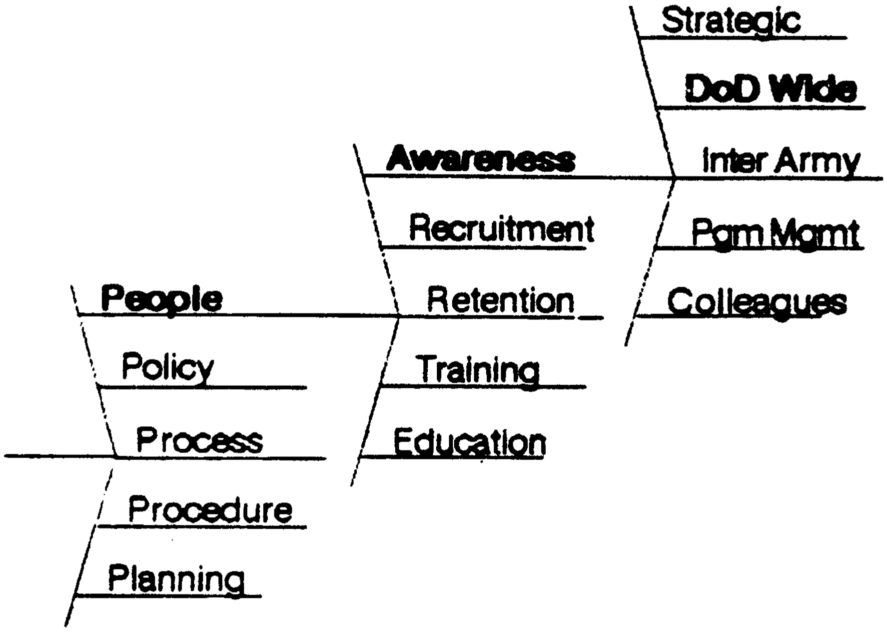
No Advocacy Plan with Congress

Lack of awareness concerning the criticality of software is pervasive in the Army, and extends from the lack of an advocacy plan with Congress; through the Office of Manpower and Budget (OMB), Office of the Secretary of Defense (OSD), and joint levels; to the Program Executive Office (PEO), and materiel development organizations. This lack of awareness is clear regarding the support for software technology development advances; it is even more clear that we have failed to educate Congressional, Department of Defense (DoD), OMB, OSD, and Department of the Army (DA) leaders (who control funding decisions) on software issues and to promote the benefits/returns of quality software investments to the Army.



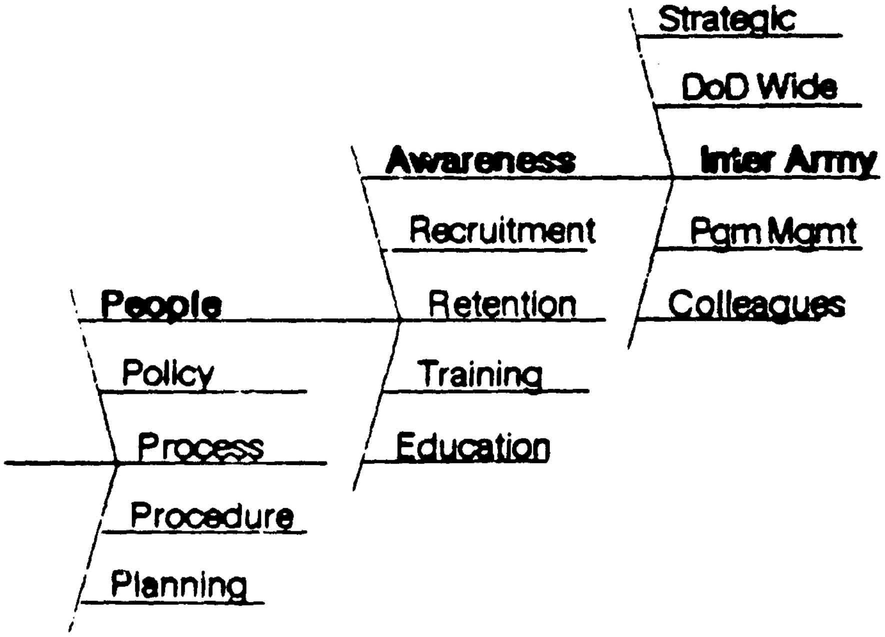
Lack of Integration/Consideration above Army Level

Naivete on the part of OSD is evident when it appears as a "doer" rather than as a policy formulator, resulting in poor coordination of critical service software policies. As an example, the Software Technology for Adaptable and Reliable Systems (STARS), although it was well intentioned, is not tied to the joint needs of the services or connected to the problems of the Army. The Joint Logistics Commanders efforts to address software issues are hampered because of limited funds and personnel resources alloted to joint activities. Problems are perpetuated by the promotion into key positions of staff without the prerequisite depth in technical software areas to be effective in improving the services' software management capability.



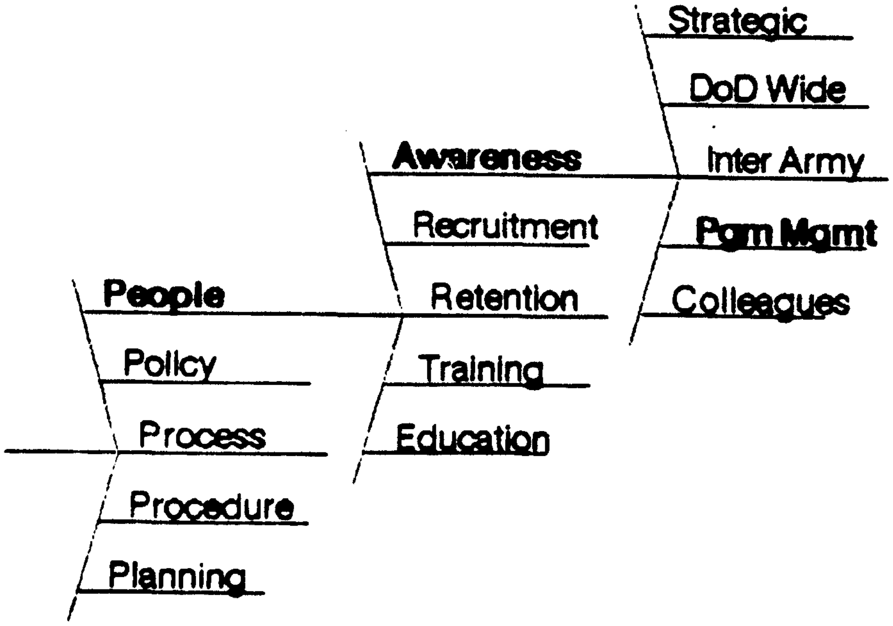
MCCR Software poorly Understood outside of AMC

Within the Army, confusion exists regarding who has central responsibility for software decisions. For example, the roles of Director of Information for Command, Control, Communication and Computers (DISC^4) and the Office of the Assistant Secretary for Research, Development, and Acquisition are overlapping regarding computer resources in system acquisitions. As a result, there is no body of expertise at the Department level to effectively address the policy and funding issues of systems with real time, embedded computer resources. General Officers of all commands and Senior Executive Service (SES) staff frequently assume their positions having matured in a non-computer-oriented world. Relatively few have subsequently acquired the level of computer expertise necessary to adequately identify problem areas in automated programs' development or operational environments, or to direct the most effective solutions.



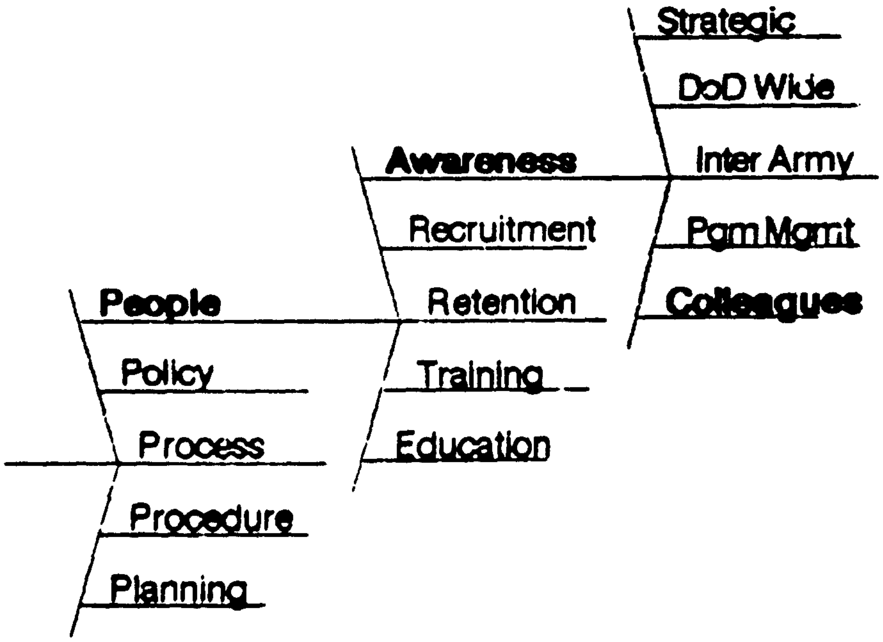
Little Software Awareness at PEO, PM, MSC

Little software expertise exists at the PEO level. Thus, quality software considerations are not engineered into a system from the concept and design phases, nor are sufficient testing time and resources allowed in development schedules. Despite the mission of the Life Cycle Software Engineering Centers (LCSECs), few PEOs tap the advisory personnel and support available to them through the LCSECs. When the Centers are utilized, frequently ‿ horizontal slice of lower skilled staff is acquired rather than a vertical slice with the varying degrees of experience necessary to properly evaluate and guide all aspects of a program's software decisions. PEOs do not plan and commit program funds for this software counsel, nor do they effectively utilize the Computer Resources Working Group (CRWG) or Computer Resources Life Cycle Management Plan (CRLCMP) concepts. These mechanisms exist to support well managed software acquisition and support, but their value is unrecognized or they are ignored as unessential.



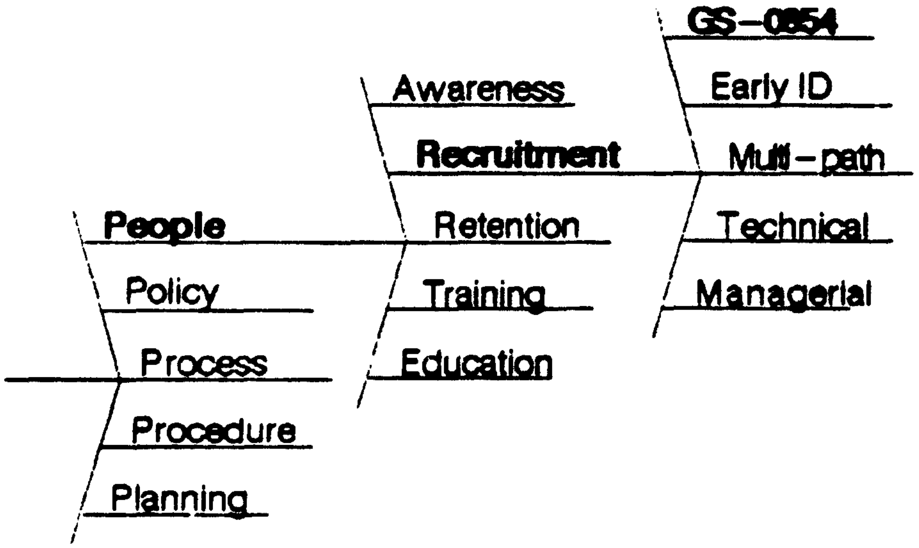
Failure to get support from PPBES/HW Colleagues

Little understanding of the Planning, Programming, Budgeting, and Execution Systems (PPBES) process and the need for software visibility exists among those in the Army Material Command (AMC) responsible for software acquisition. Likewise, little understanding of automation concerns exists in the Army among those responsible for PPBES activities. Without software staff well versed in budget request and review procedures, the needs of hardware invariably override software funding reeds. This condition extends to spaces as well as to dollars and consequently forces the contracting of software work because Army facilities are understaffed or underskilled to perform the work in-house. The intent of retaining the combined software/system expertise in-house at the LCSECs is thus frustrated. We have sacrificed badly needed professional slots by retaining slots for service/maintenance-oriented functions, which could more readily be contracted without causing stagnation in the growth of our technical capability.



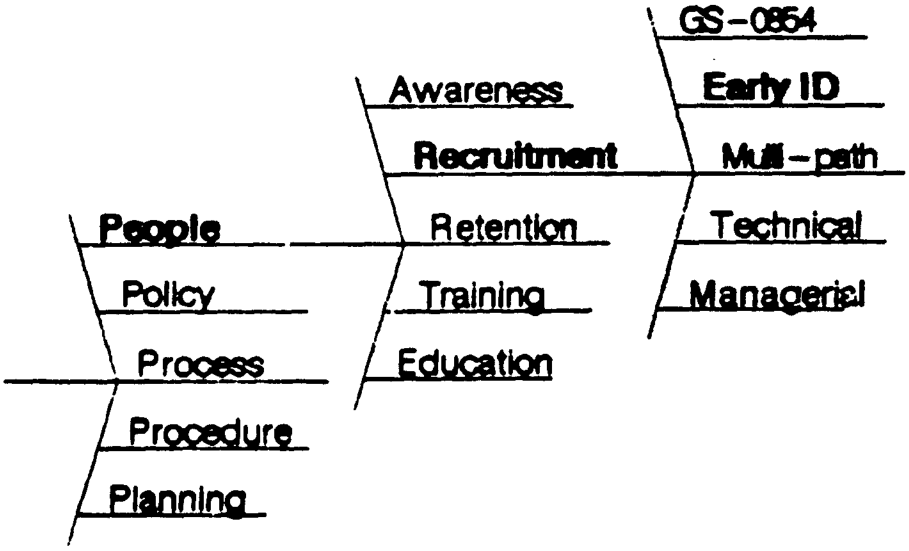
No Capitalization of new Computer Engineer Series

Although the Computer Engineer job series (GS-0854) has recently been created, little recognition has been given to the fact that actually acquiring these staff will be difficult. This is due partially to the stringency of the series' qualifications (See Position Classification Standard for Computer Engineering Series GS-854 (TS-83, January 1988)) and partially to the lack of a crossover program allowing current government staff to acquire credentials necessary to transition to the new series. Further, there does not appear to be career management assistance, either formally as a personnel service or informally using a counselor/mentor approach, to guide personnel in identifying and selecting positions that will foster their technical growth in software engineering. Having no definitive career path with recognized, required steps inhibits steady progress toward developing the Army's required software engineering expertise.



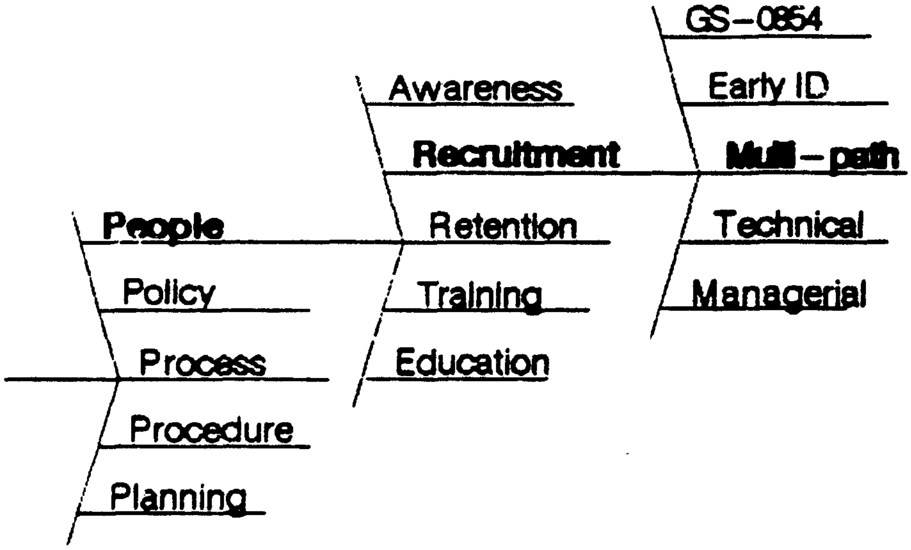
Best Students lost to Industry early in College

With lower entry level government salaries, the Army's ability to hire the best computer science graduates away from higher paying industry opportunities is severely weakened. Entry level government salaries of $25,000/yr put Army recruiters at a great disadvantage in competing with industrial offers of $30-40,000/yr. The differential is clear. Also, because of their concentrated efforts to attract college students throughout the undergraduate college years, industrial recruiters are far more successful in signing up the best students well before graduation. Conversely, the current Army co-op program does not successfully cultivate students by pre-selling an Army career, nor by tieing the co-op program to a geographically oriented intern recruiting program. In many instances, we have failed to permanently employ the co-ops we have had, either because of poor co-op - permanent position transition planning, or by failing to create the stimulating technical environment required to challenge new, eager graduates, valuable staff continuity between co-op experience and long term employees is thus lost.



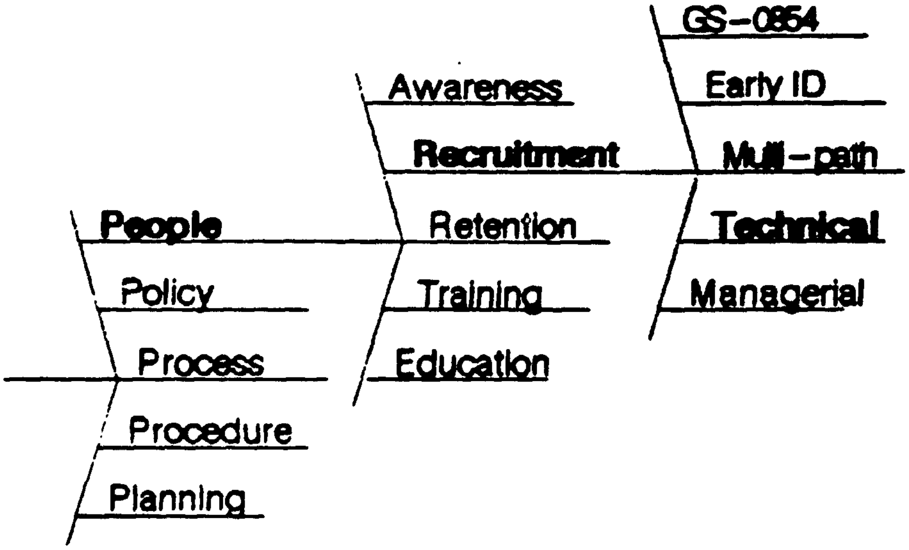
Failure to Structure multi – path Career Program

Although effective managers must have a balanced mix of technical skills and management experience, technical career growth is sacrificed for the more attractive promotional opportunities of the management track. By limiting the technical growth of our management staff, we necessarily deprive those tasked with decision responsibility from obtaining the depth of understanding their positions require to make those decisions. Further, little if any preparation is provided to move from the technical path to management responsibilities.



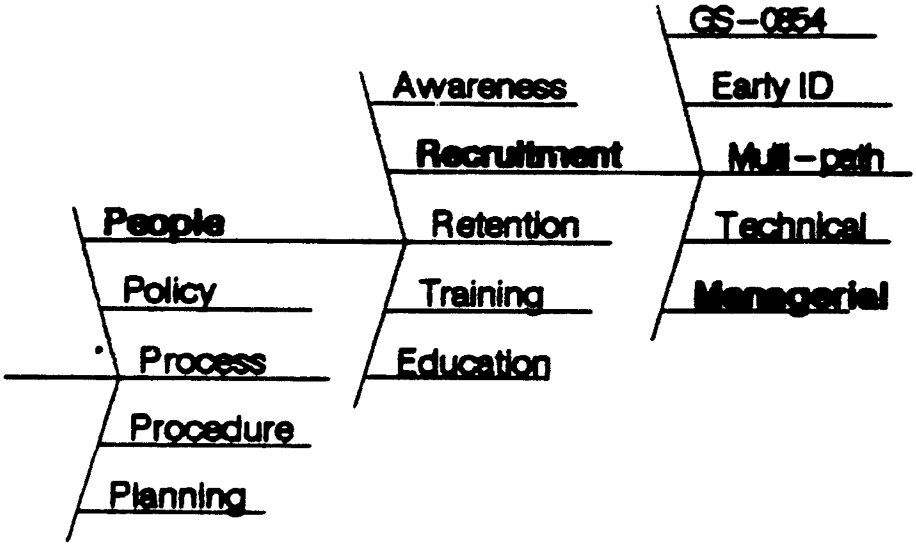
No Effective Technical Development Program

For those electing to remain in the technical arena, training courses do not necessarily track to a clear career development scheme. Disillusionment also frequently results when staff are not utilized in their expressed area of interest (i.e., when engineers are not used as engineers). Government employment also offers less incentives than industry to obtain advanced degrees, an important recruiting advantage and career enhancer for computer science graduates. Considering that in these fast paced technology areas computer science and engineering knowledge has a half-life of approximately four years, continued educational/ degree opportunities are high priority items for candidates weighing employment options.



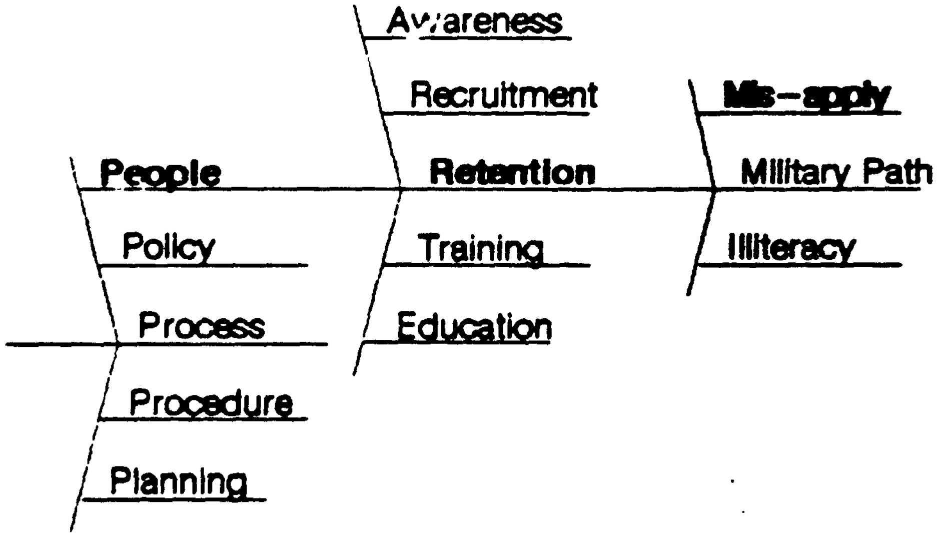
Inadequate Management Development Process

Management career development is equally unstructured. Opportunities are not provided for managers to continue association with the technologies of their programs and with the development of management skills. These opportunities are necessary to properly plan, schedule, direct, and evaluate their programs' progress. Only those familiar with technology advances can effectively write and monitor contracts so that the best products can be specified and acquired for the Army. Effort must be made to turn Army managers into "smart buyers" and increase practical technical involvement to attract the strongest leaders to Government service.



Talents and Education mis-applied

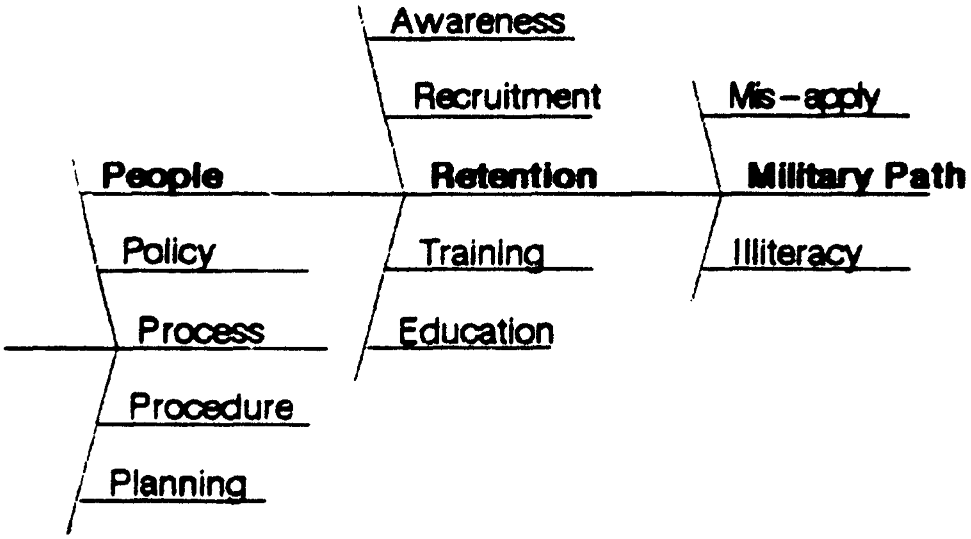
The issues surrounding retention are not unlike those affecting recruitment. They are, however, intensified because not only does the Army loose the staff, it also looses the benefit of the experience they have acquired at Government/military expense. Engineers frequently are assigned to non-engineering work, and mid-level staff consume most of their time responding to bureaucratic requirements rather than to hands-on technical assignments. Recent graduates often revert to industry to get the more challenging positions they need to develop their technical capabilities.



Lack of Career Path for Military SW Experts

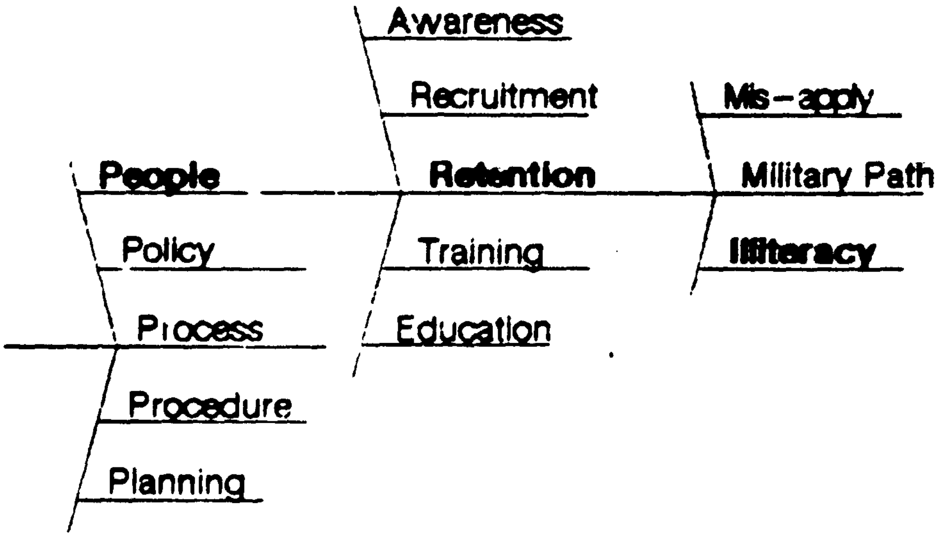
Some of the greatest drains are seen in the military personnel area. Even with strong credentials and performance, promotional opportunities have been sorely lacking for officers and enlisted assigned to software-oriented billets. This pattern sends blatant, negative messages and deters younger officers who might become valuable assets from moving into software programs. Instead, they choose to leave the Army and follow other, more career enhancing avenues.

A contradiction is inherent in this pattern that has not been lost on industry, even if it is not apparent to the Army. Disillusionment with frustrated career progression makes mid-level military with a vital combination of field experience and academic or practical software talents ripe for industry offers. With so few "software smart" soldiers in the field, those that do exist are immediately identified by industry as key assets and lured away by impressive salaries and advancement opportunities.



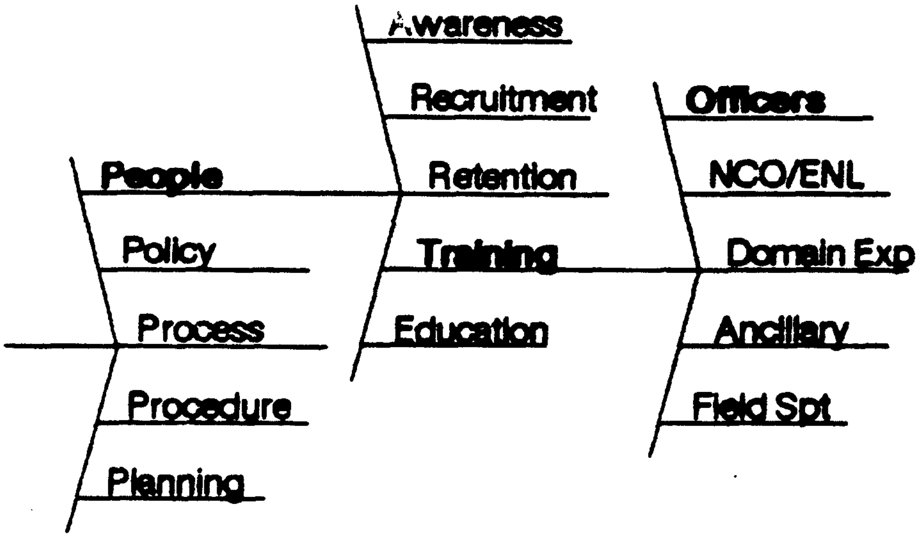
Army Leadership Illiterate about Embedded SW

A consequence of the mid-level military drain is that the goal of developing the required body of software knowledge in our senior Army leadership is becoming more and more unattainable. We are losing the young officers who would eventually assume those positions. Without advocates at the higher policy, funding, and prioritization levels of Army decision making, automated systems, software technology, and staff development programs will continue to suffer from naive decisions and funding deficiencies. Other services promote software awareness in their senior ranks by having General Officers presenting software literacy/information programs. Where attempts at information programs have been made in the Army, they have been conducted by lower level staff with little or no influence to affect constructive changes. By relegating such important work to lower levels, we perpetuate a "software second" mentality and delay effective action to deal with software issues.



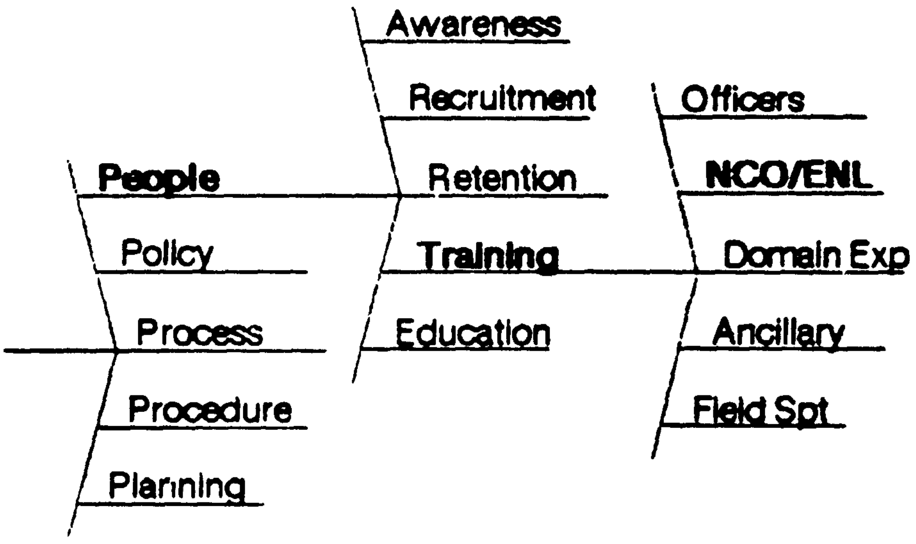
Inadequate Officer Training on Software Role

Although multiple formal training courses are provided for nearly all aspects of an Army officer's professional development, very few opportunities exist for mission critical software and computer resources training. Officers are consequently ill-equipped to direct and oversee computer resources operations in the field. What training does eyist (e.g., short term assignments to the LCSECs) is generally limited to junior level oficers; thus, those with decision making responsibility are prevented from acquiring sufficient knowledge to make informed decisions.



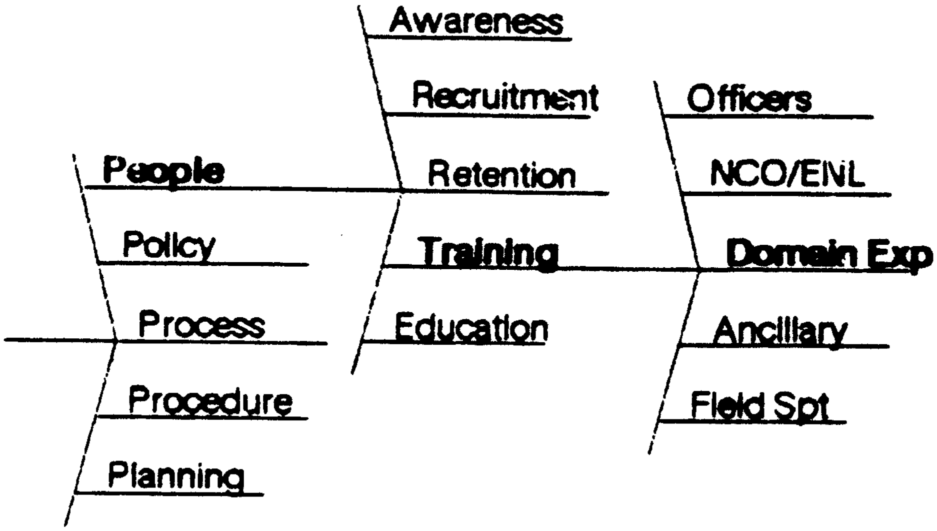
Little Infusion of Software Literacy for NCO/ENL

Little or no advanced software technology infusion exists at the Non-Commissioned Officer (NCO) and enlisted levels (ENL) (courses are still limited to COBOL), yet these are the people responsible to configure, operate, and trouble-shoot complex automated systems in the field. Absence of the vital connection between the application area/weapons system knowledge and computer resources knowledge further prevents those trained in field operations from recognizing and reporting software problems or those with only the software background from recognizing the perspective of the soldier in the field. We fail to recognize a great asset in NCO as experienced field users, and to assign them as advisors to PEOs/PMs during the development of new systems. Yet those few trained in both arenas exhibit the highest commitment to providing comprehensive field support while ensuring that the software integrity remains intact.



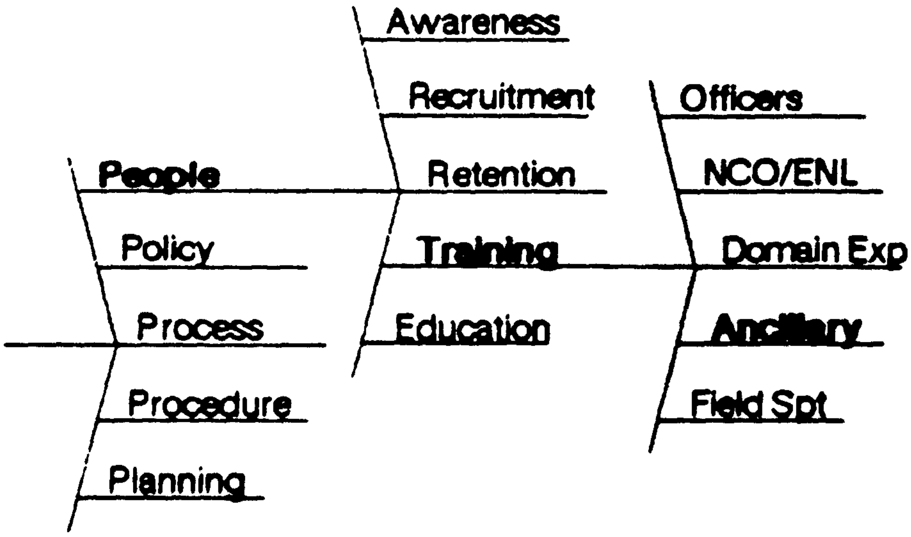
Poor Weapon System Knowledge by SW Experts

In the same way that operational field officers, NCO, and ENL are ill-equipped and trained to deal with software issues, software developers frequently lack the field user's perspective, understanding, and expertise necessary to analyze a new system's requirements, implement it, and thoroughly test it at completion. One must either be a good communicator, missile operator, or artillery controller to develop a truly responsive automated system, or be diligent in researching a given field with the assistance of qualified domain experts. Current development programs focus only on the software, not on the maturity of the developing staff's understanding of the application domain.



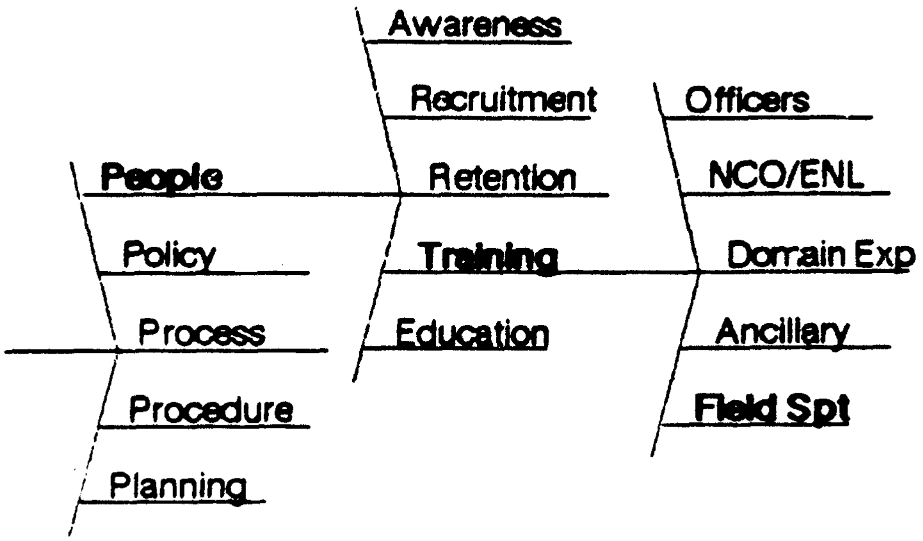
General Lack of Software Ancillary Skills

If training in the development and operational concepts of automated systems is lacking, training in such ancillary computer resources skills as Interoperability, Quality Assurance (QA), and Configuration Management (CM) are non-existent. Yet these areas, along with knowledge of proper installation and back-up procedures, proper environmental conditions and preparation of systems for transport; and overall computer system management (e.g., operating system upgrade, new software capability deliveries, and recovery from system crash) are also crucial skills not being addressed.



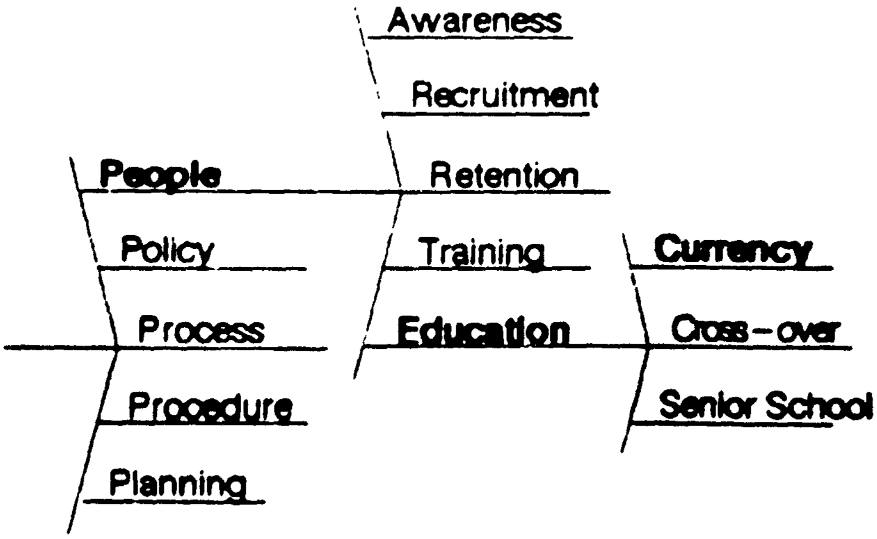
Poor Development of Field Software Analysts

Currently no program exists to develop qualified field software analysts. Because a cross-discipline capability incorporating software engineering has not been realized, training for warrants, E5s, and E6s with field experience has not been oriented to software analysis. A valuable link from the user in the field to the LCSECs responsible for the computer systems support is thus being lost. Field software analysts must be experienced in many systems in order to be effective. While obtaining this broad range of system knowledge is recognizably difficult, we have failed to use existing resources such as the LCSECs, where systems with similar functionality are assigned, as a training environment for field analysts.



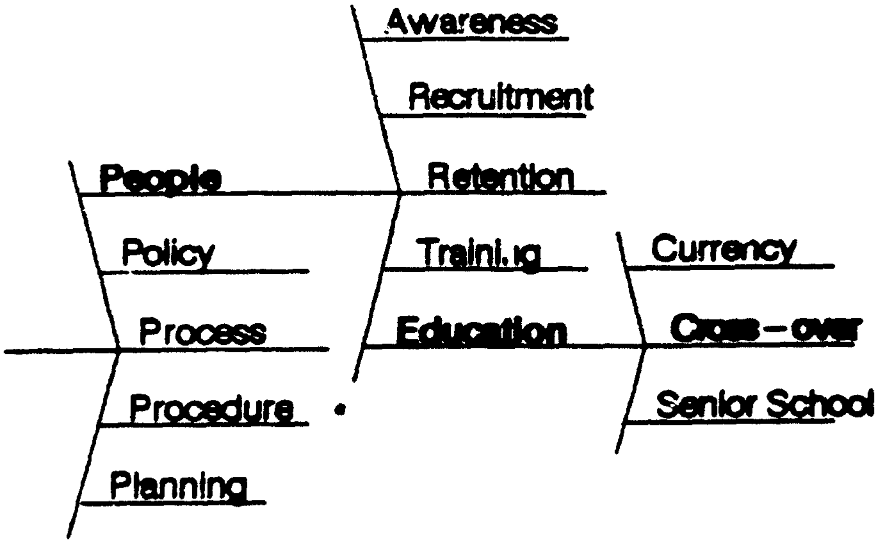
Failure to keep Software Professionals Current

A lack of educational opportunities is a deficiency whose implications affecting recruitment and retention have already been cited. The competency level of staff who do remain in government positions is also jeopardized by our failure to keep software professionals current in technology advances. While a few full-time degree opportunities as well as part-time night programs are available, the single course approach (a good option for subspecialty development) is underutilized. This may be because of a lack of commitment to educational funding, or perhaps because of a lack of a mentor or guidance program that encourages staff currently working in automation- related fields to acquire the added skills to cross-over into software engineering. Education is also lacking for those tasked with preparing the essential software portions of RFPs for new procurements, e.g., writing clear, concise, and complete requirements specifications or stipulating the responsibilities and interactions between prime and IV&V contractors. Without proper education, government and military staff are also ill-prepared to critically assess the quality of software systems being delivered by contractors; formal reviews are of little value if the reviewers do not have the required technical foundation for evaluating deliverables and services.



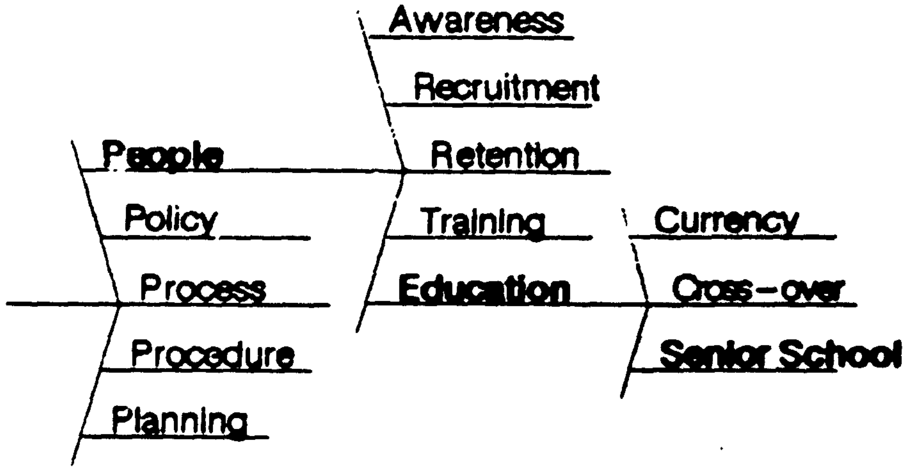
No Program to Provide Cross – Over Education

Because of salary and technical considerations already cited, we cannot recruit enough software engineers directly out of college. At the current time, no cross-over program exists in the Army to take engineers from other areas or former military with valuable field experience and retrain them in the critical software engineering disciplines. By doing without a cross-over education program, we perpetuate the already critical lack of software expertise in our government staff.



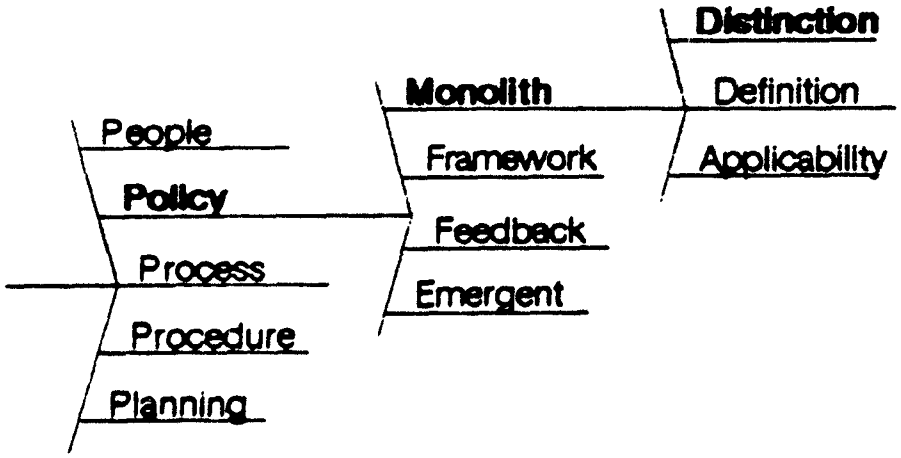
Little Emphasis on Software at Military Schools

An obvious deficiency is that little or no software emphasis is present in the senior military schools, a failing that promulgates the "software second" mentality and further delays getting the senior leadership in touch with the criticality of software to the services. Even at the Defense Systems Management College (DSMC), little education is provided in the skills necessary to direct an automated program. Inadequately addressing the area of software management ignores an aspect of a program with potentially the greatest risk for preventing the timely cost-effective delivery and correct field operation of a complex weapons system.



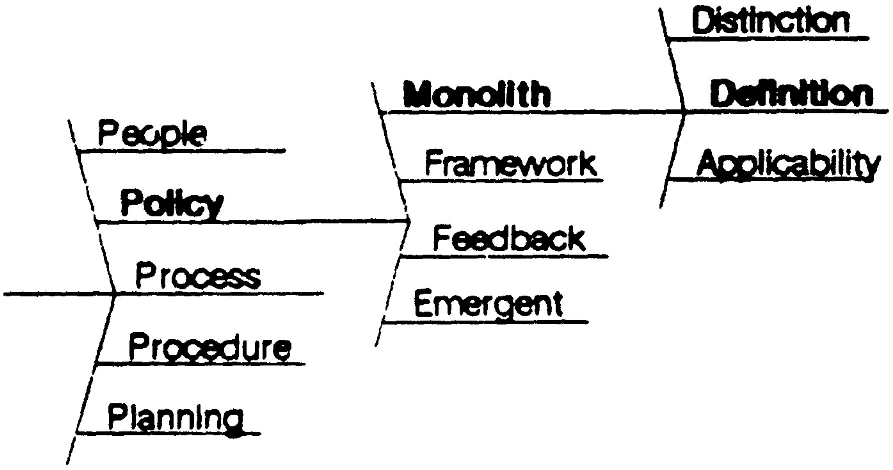
No Distinction of Embedded Software Peculiarities

DoD acquisition policy pertaining to computer resources is based, in part, on public law. With the enactment of the Brooks Bill (Public law 89-306) in October 1965, the General Services Administration (GSA) became responsible for the economic and efficient purchase, lease, maintenance, operation, and utilization of automated data processing equipment by the federal departments and agencies. The distinction was made between general purpose business, financial type applications, and the special purpose DoD weapons systems applications of computers. This latter category of computer systems was exempt from the provision of the Brooks Act. The Warner Nunn Amendment broadened the class of computer resources and exempted them from the Brooks Bill and was the genesis of the new term, Mission Critical Computer Resources (MCCR). However, the distinction of MCCR has not been codified into regulations, standards, and pamphlets. This absence of a systematically arranged and comprehensively collected set of procedures to provide a distinction for embedded systems and their peculiarities has resulted in a broke system, before it was created, in software acquisition.



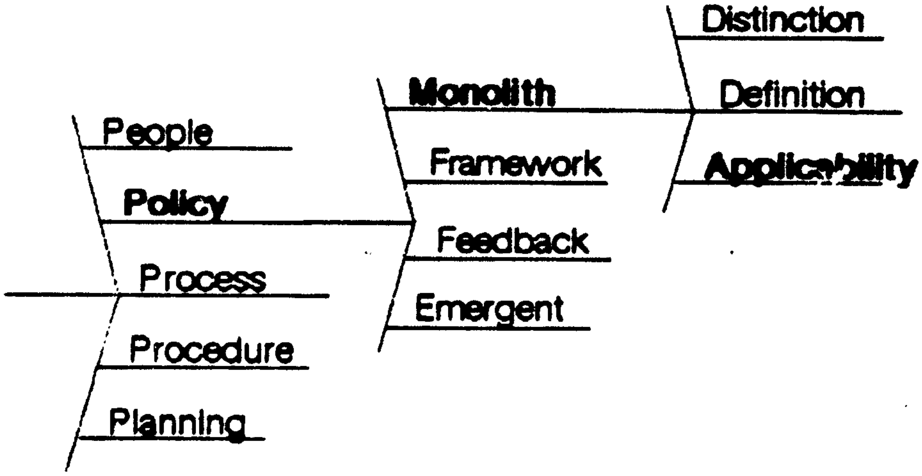
Confusing, Overlapping Areas of Responsibility

While MCCR were developed and acquired under policies and procedures outlined in DoD Directives 500C.1, 5000.2, and 500.3, the Army directed the use of Army Regulation (AR) 70-1, "System Acquisition Policy and Procedures," a hardware only regulation. When OSD shifted oversight of life cycle management for MCCR by expanding the definition of general purpose automated data processing to include the Army Systems Acquisition Review Council (ASARC) and Joint Requirements Management Board (JRMB) guidelines established in DoD Directive (DoDD) 5000.1, the systems fell to review by the Major Automated Information Systems Review Council (MAISRC). There has not been established a focal point for advocacy, at DA level, for mission critical, embedded system software. It must be recognized that mission critical systems are different from Automated Data Processing/Management Information Systems (ADP/MIS) and equal emphasis must be provided for acquisition policy, career development, and budget advocacy for mission critical software.



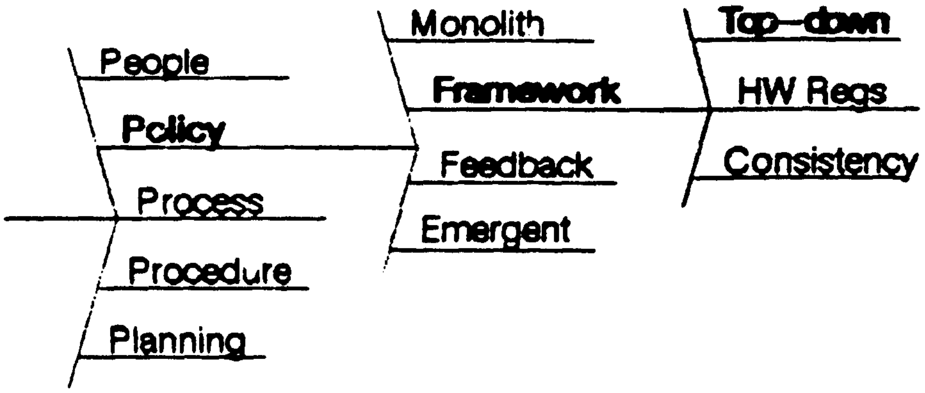
Application of AR 70 vs AR 25 Regulations Unclear

The Army does not have one general policy defining an embedded tactical software framework. Numerous regulations and pamphlets have been promulgated that cover most of the subsets of the software life cycle process. However, neither the PM, PEO, nor the Army Acquisition Executive (AAE) can turn to any one policy document to determine what constitutes the life cycle software acquisition process. For example, in AR 70-1 references to hardware occur four times more frequently than references to software. There is also no mention of the need to allow capitalization of software, that is, the notion of including in the contractor's assets his ability to produce software. Life cycle acquisition of embedded tactical software systems must be done through two different policy channels. Development is done under AR 70-1 (with NO supplemental AMC regulation).



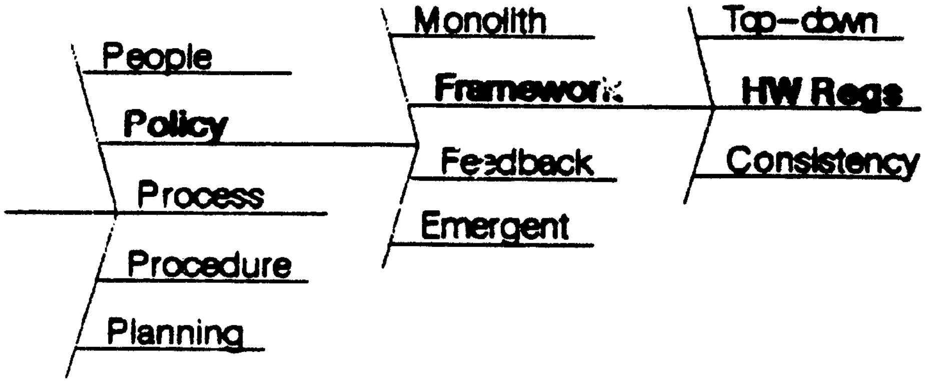
No Top – down Plan to Address Software Policy

Software for embedded systems has never been acquired on a life cycle basis under one policy stream. The approach taken by DA was to "force-fit" software into hardware regulations with no consideration for its peculiarities. This failure to adequately address software in Army regulations and pamphlets left a void that will have a costly impact for PDSS well into the 21st century. There are many regulations that address (or do not address) software in various disciplines; however, there is little or no consistency between these regulations as to how software should be acquired and sustained. A major problem created by this inconsistency is that there is no standard for embedded computer resources that generate ever spiraling costs.



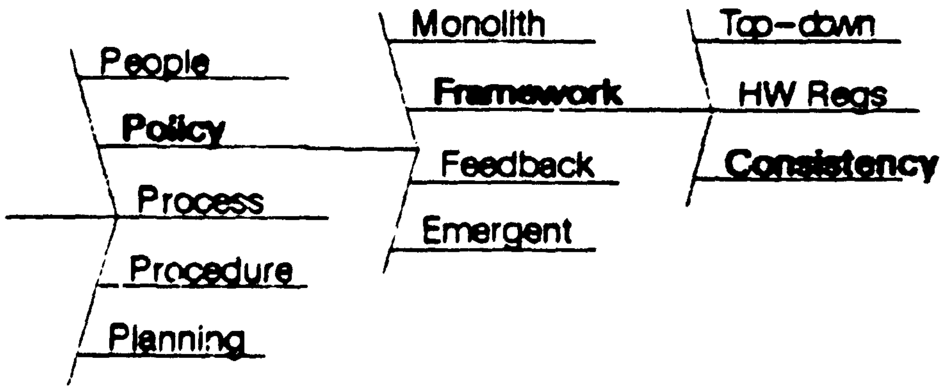
Failure to Address Software in Regulations

Because AR 70-1 did not adequately address the life cycle support for embedded software, AR 70-XX was prepared to implement the DoD directive. It has been in draft form for several years because of a disagreement between Headquarters and DA activities over the proponency for embedded systems. This lack of Army policies and procedures for embedded systems has negatively affected the development and support of software over the past 12 years. During this time, tactical software programming languages proliferated and software supportability was not built into many systems. Consequently, the Army does not have the means to effectively and economically support much of its tactical software.



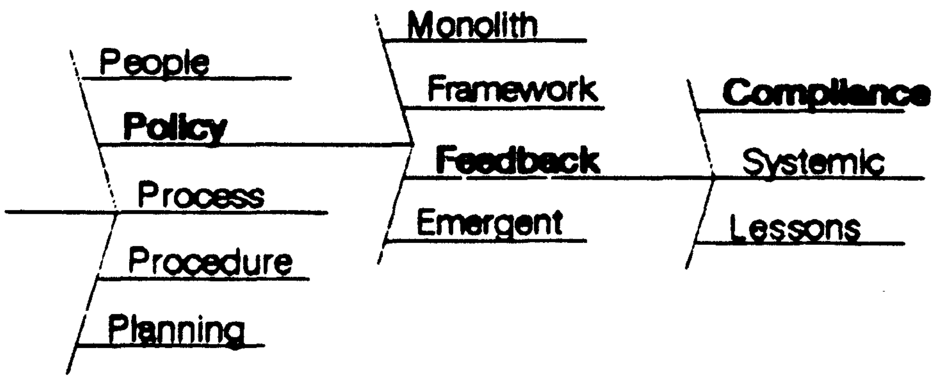
Lack of Consistency between Regulations

A lack of consistency between regulations and standards generated ingrained developmental problems that will cause a continued spiraling effect in support cost for another 15 years. This lack of consistency failed to provide the means to adequately build software supportability into most embedded systems because the Army did not have sufficient oversight for supportability. The root cause of inadequate supportability was that planning (lack of standards promulgated by regulations) for software support either was not accomplished or was not adequate. As byproducts of poor software planning, adequate software documentation and software support environments were not acquired. The recent emergence of DoD Standard 2167A, "Defense System Software Development," and DoD 1467 (AR), "Military Standard Software Support Environment," will eventually combat these problems. However, neither AR 70-1 or AR 25-1 recognizes these critically needed standards for directing the effective and economical support for embedded software. Without adequate software documentation and support environments, the Army does not possess the means to change tactical software to keep pace with new doctrines for threats, to enhance system effectiveness, or to correct system deficiencies.



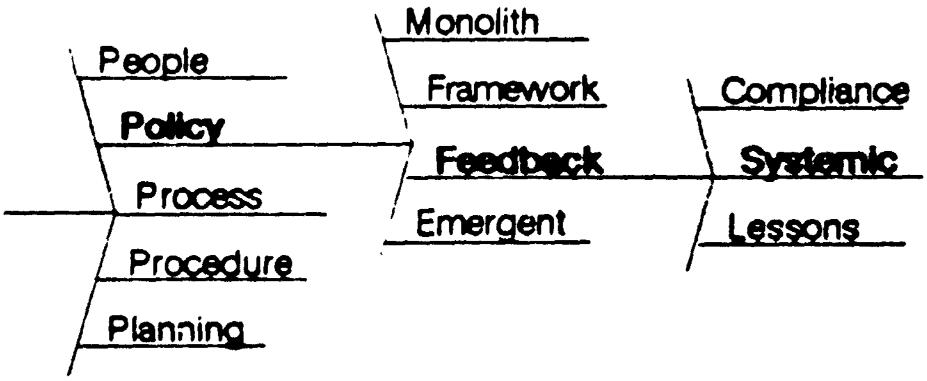
No Process to Force Regulatory Compliance

The Army has not issued regulatory guidance to implement DoDD 5000.29 for the management and control of embedded tactical systems. Lack of Army policies and procedures has created a void for the implementation of a process to enforce and/or review regulatory compliance. Without a review process, proliferation of tactical software has been unchecked, and software supportability has not been built into embedded tactical systems. Had the Army implemented DoDD 5000.29 and its policies, there would have been a framework to create organizational structure and systematize procedure and review compliance. Unfortunately, this organization has never been created because of conflicting regulations and the basic lack of embedded software advocacy at the DA level with the requisite authority. As a result, no oversight system has been established at the Major Command (MACOM) level for computer resources planning, software language standardization, configuration management, and software supportability.



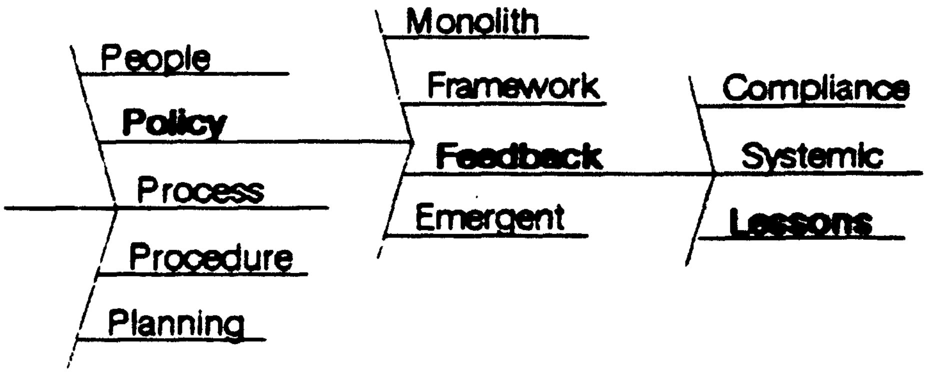
Failure to Identify Systemic Problems

Failure to identify systemic problems was a negative fallout from the lack of the aforementioned process. Computer Resources Management Plans (CRMP) were not prepared for many embedded systems, which usually resulted in the after the fact efforts to create the essential software support environments before a new embedded system was fielded. CRWGs were also not established for most embedded systems resulting in no effective monitoring of the systems' software development. Many of the quality factors critical to maintenance were not built into the software system; therefore, the correction of refinements and flexibility of enhancement were costly or impossible without total redevelopment. Inadequate software documentation was also a situation in the majority of many embedded systems. As a consequence, clear details about the instructions and definitions that make up computer programs and thorough information on computer program functions, capabilities, and operations were not available for software support. These systemic problems were never identified until the later 1980s when system software maintenance costs became so high that the programming and budgeting processes received substantial dollar cuts because there was no reasonable justification for the high costs.



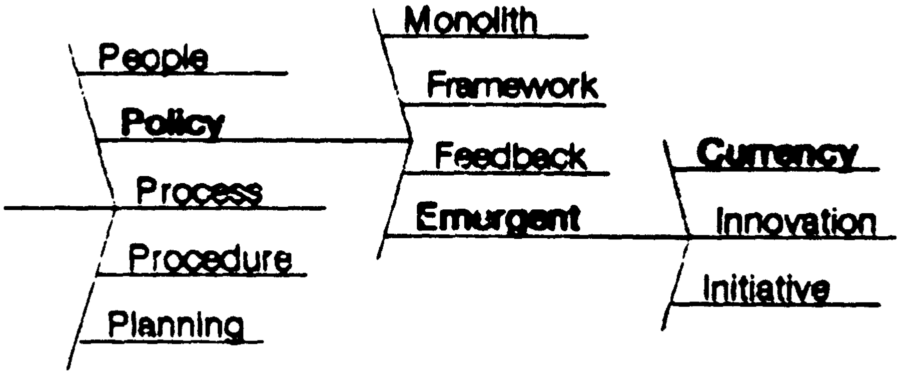
Lessons Learned not Passed On

Because there had been no means of identifying systemic problems, any lessons learned on how to affect viable software policy generally remained at the lowest acquisition or support level. When a PM learned that a CRWG could generate a functional CRMP that resulted in positive support for an embedded system, it remained locked within the closed loop of the PM or software support center. There were no lessons learned for feedback to other PMs that were not performing adequate development of computer resources; therefore, no realistic audit trail as to the insight of software development was established.



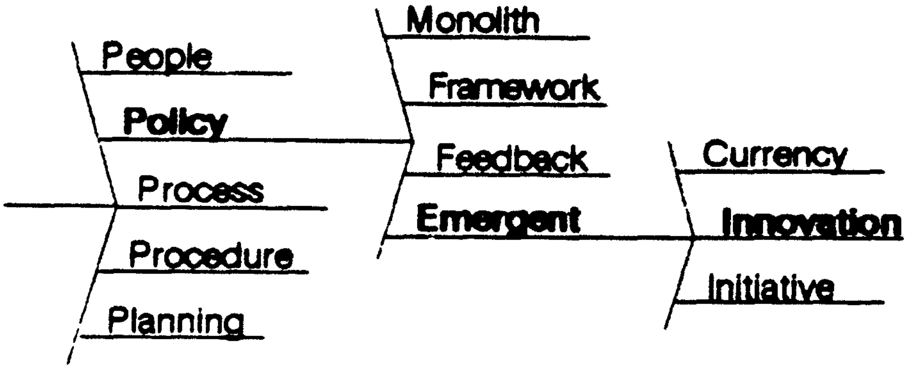
Policy not Current with Latest Technology

Progress to meet sophisticated software technology thrusts through viable Army policy is essentially non-existent. Currency of policy or the lack thereof has been the most prominent reason for uncontrolled computer language and microprocessor proliferation. One example of no existing policy for emerging technology is the total lack of Programmable Read-Onby Memory (PROM) and Erasable Programmable Read- Only Memory (EPROM) technology insertion into development of new embedded systems. Another is the lack of reuse of software components among concurrent developments. An evaluation of the Advanced Field Artillery Tactical Data System (AFATDS) and Maneuver Control System (MCS) Ada software developments revealed that reuse of components between the two similar (command and control) systems was nonexistent. Policies are not current to direct this type of technology insertion/innovation into the contracting and proposal evaluation process.



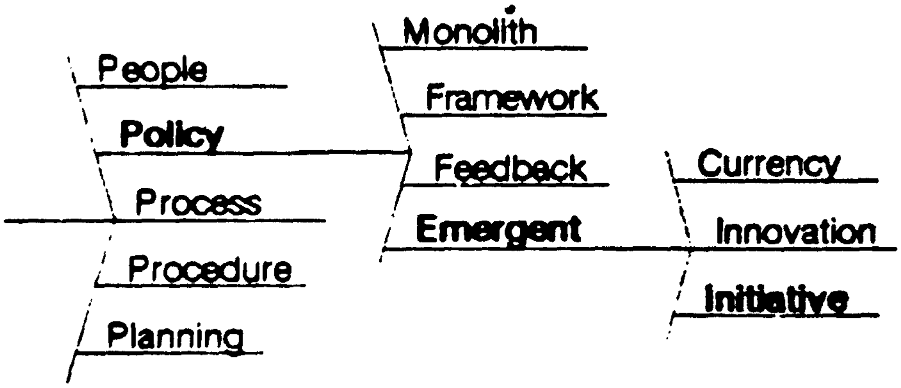
Regulations Hamper Technological Innovation

On the whole most regulations pertaining to computer hardware and software acquisition tend to prohibit or omit the encouragement of technical innovation. The need for a generic software program points out this regulation void. Generic software from one view point might create both a technical base that addresses technology for tomorrow and an engineering base that supports today's problems. This concept is certainly not a rote application of technology but an initiative that would span many systems; thus providing for a reduction in developmental and sustainment costs. An apparent problem that exists is the embedding of technology in policy. One such example of this is DoD-STD-2167A, which places the traditional waterfall approach to software development as an obstacle to innovation in a technological sense.



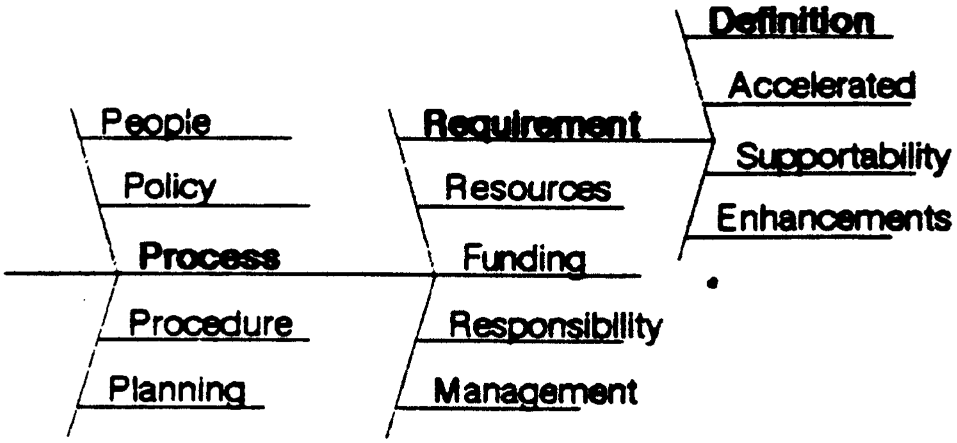
Rote Application of Policy Hampers Initiative

An embedded system should be predicated on the state-of-the-art technology only when the benefits of the new technology offset the accompanying risks. This principle is easy to state, but it is hard to apply because of the difficulty in getting reliable information with which to assess the trade-off of risks and benefits. The only consistently reliable means of getting such information is by building prototypes that embody the new technology. Prototyping, either at the system or critical subsystem level, should be done for all new system starts. Operational tests should be combined with developmental tests of the prototype to uncover operational and technical deficiencies before a decision is made to proceed with development. At present there is no policy or regulation that addresses prototyping and testing in the early stages of development, but does not present obstacles that prohibit technological innovation.



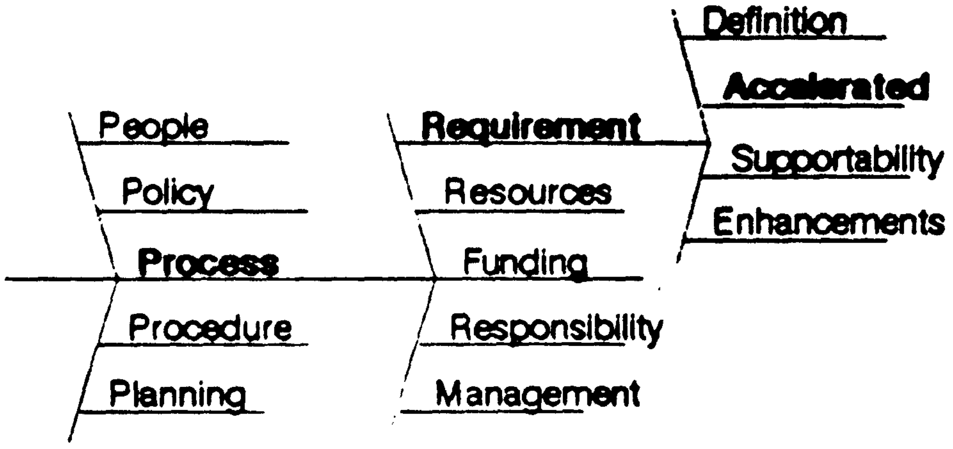
Requirements Cannot be Fully Defined Early

As stated by the 1983 Army Science Board Study on Acquiring Army Software, "the most significant cost and schedule growth drivers for software are requirements and specifications changes." Changes are inevitable and the rate of change will increase based on the enemy's technology explosion and the stability of the fiscal program for a system based on its perceived, not actual, DA priority. Rapid prototyping, for all its hype, is rarely a reality that can effectively validate functional requirements. "A" and "B" specifications usually become the casualty of a program cut, which leads to inaccurate design, testability, and last, but most costly, supportability.



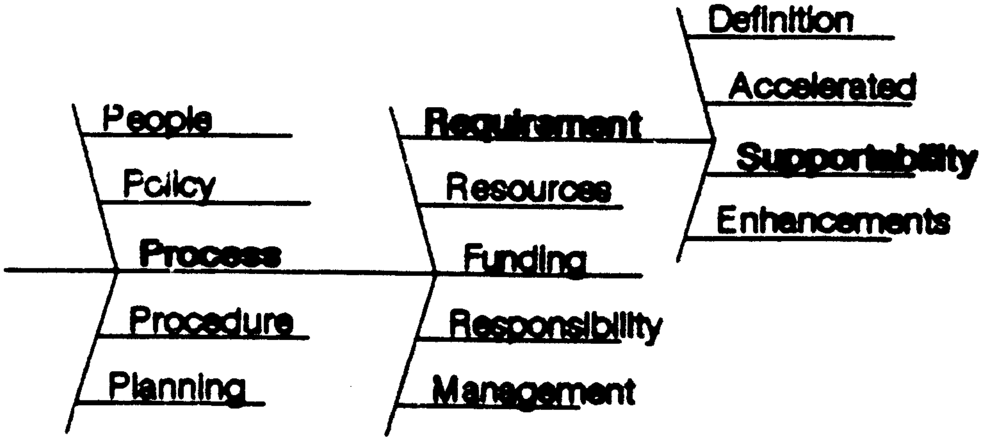
Schedule Driven Software Development Process

Poor initial software requirements definition and subsequent requirements changes usually result in cost escalations and schedule delays. The enforcement of disciplined software requirements, change control, and configuration management after requirements definition rarely recults in a stable functional software baseline. When a development falls behind schedule, all management focus is placed on code generation. Usually the first software product to be cut is documentation relating to design, code, and integration. Testing suddenly is only viewed at the system level, with no test or integration of modules or packages. Without quality of design, realistic test plans cannot be generated, and a system may be fielded that does not meet operational requirements. This then drives the requirements for Other Procurements Army (OPA) and Operations and Maintenance, Army (OMA) dollars well above the programmed level for planned sustainment.



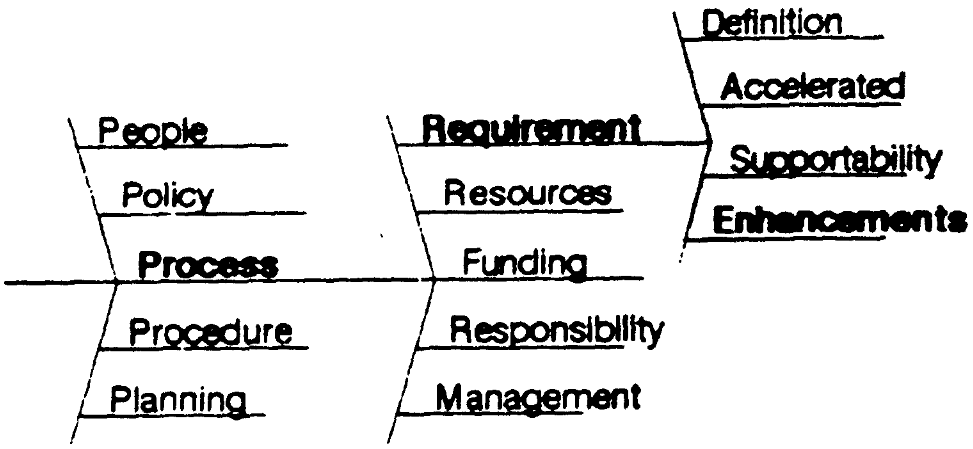
Need for Supportability Rarely Considered

Once the software development process becomes schedule driven, chances become low with regard to retaining the necessary programmed dollars for adequate software documentation and software support environments. This in turn consumes a disproportionate percentage of resources, thereby preventing allocation of sufficient resources to new systems to ensure that they will be supportable when fielded. This domino effect then increases the cost of refinements and enhancements since much of the software has to be reverse engineered.



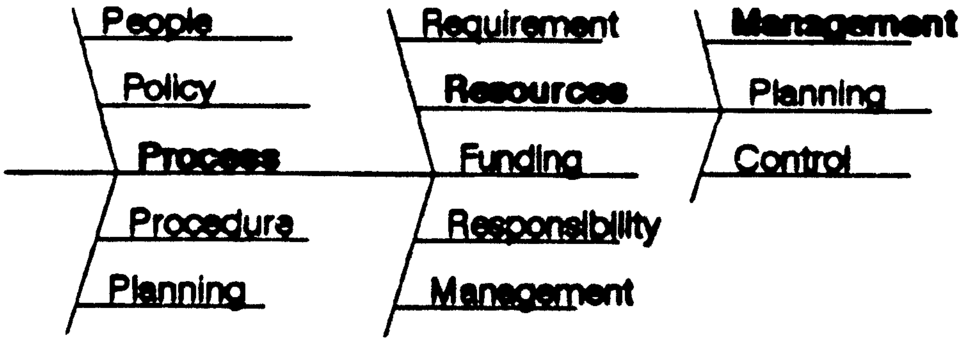
Enhancement/Modification Process Uncontrolled

With no requirements stabilization, system design usually fails to anticipate need for growth. Another result of uncontrolled modifications is the lack of interoperability. This has had a drastic effect on the electronic ability to command and control the battlefield. Few systems have the ability to intraoperate within their functional node, and certainly only a limited capability exists to interoperate between functional nodes. All of these issues relate to the policy problems; they are a fault of NO acquisition methodology within the Army.



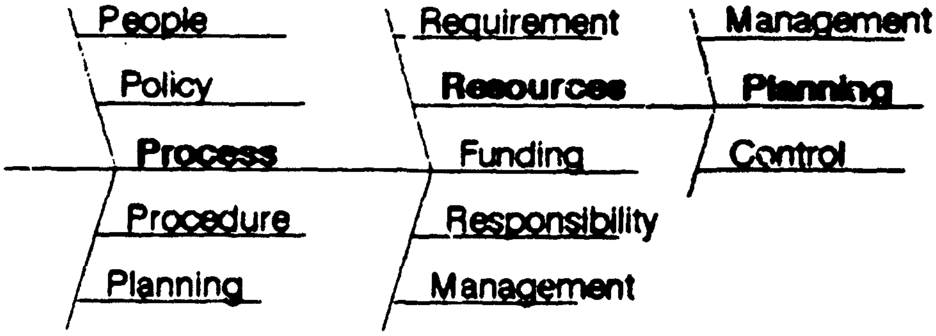
CRMP Ineffective for Program Management

Planning for the software support of automated tactical systems is not adequate. CRMPs (plans for the life-cycle support of automated tactical systems and related software) often are not prepared. Computer plans that were prepared generally lacked sufficient details and in many instances were simply boilerplate in nature. CRMP have not been prepared in a timely manner; thus uncoupling the plan from the PM's acquisition strategy. The prescribed format and content for CRMPs encourages them to be composed of boilerplate in a voluminous manner and contain little strategy with superfluous and irrelevant material. The CRMPs are essentially stand alone documents not tied to acquisition strategy planning. They do not consider the incorporation of various matrix functions such as quality assurance, configuration management, or RAM. They are also not suitable as a management tool because they omit critical strategic considerations such as data rights, contractor vs. in-house support, and funding and manpower requirements. All of this produces an air of "what good are CRMP", which results in no impact toward a successful development or sustainment.



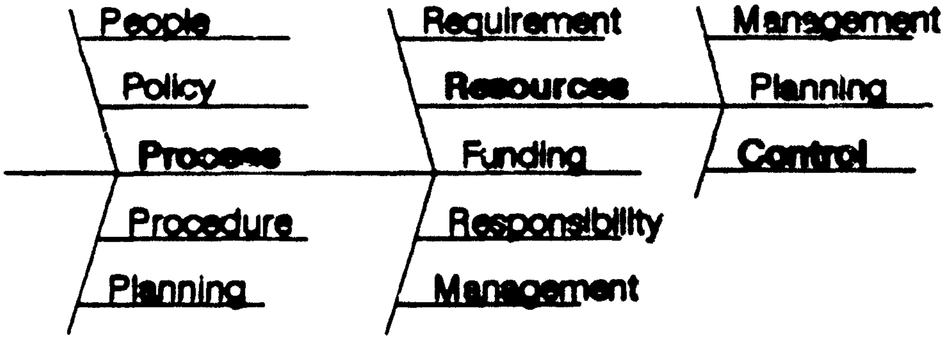
No Macro Level Capability for Resource Planning

The prime essence of the CRMP is to serve as a planning document for computer resource acquisition and sustainment. There is no funding mechanism for top-down or bottom-up program requirements that support the CRMP. Funding data is rarely visible within the appropriation line for system software. Funding data to provide requisite fiscal visibility at the DA level is possible if adequate hardware and software data were spelled out in the CRMP. To become a dynamic document it must incorpate not only a currency in technology and timing with system develcpment, but also a total coverage of all the facets necessary to make it a viable document.



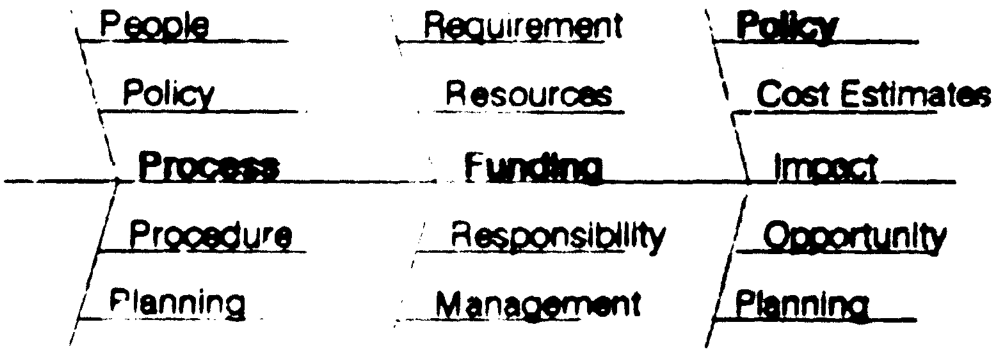
CRMP Process Totally Out-of-control

CRWGs (groups that meet to ensure that the policies, procedures, plans, and standards established for automated tactical systems and related software were followed) generally have not been established or established when required. Policy on software support planning at the DA level is vague; Army regulations mention computer plans and working groups but contain few details on their nature and use. Controls at the AMC have not been effective in getting materiel developers to prepare detailed and prompt computer plans and to establish working groups. This has resulted in inadequate software support environments among automated tactical systems. Thus, software supportability is not built into many systems, and the Army cannot cost-effectively support tactical software or readily change it during a conflict. One prime example of the control problem has been Ada waivers. Many systems went into full scale development without Ada well after the prescribed date for standardization of the language. When CRMPs are prepared, many times they have been done after development is well underway and after most critical software decisions have been made by default. This ex post facto situation resulted in greater proliferation of computer resources, which drove costs even higher.



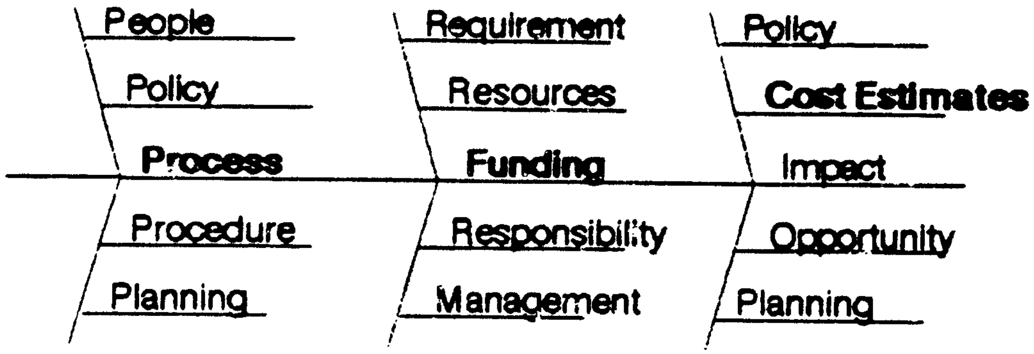
Inconsistent Funding Policy

Funding for life cycle software support has also developed into an inexacting science. Although there is a well defined way as to how software is developed, the challenge lays in what the level of funds must be in a given phase of the life cycle. The level of funding in each phase of the life cycle for an embedded system may well affect the quality of the software. If dollars are not sufficient during the development phase, resultant software be low in quality, thus impacting its reliability. All of the uncertainty surrounding software production, it's value, and how many resources are needed to develop and sustain it has resulted in a highly unstable fiscal program. In the last eight years funding policy has changed five times at the DA level with different interpretations at the MACOM/Materiel System Computer (MSC) levels. The Program Objective Memorandum (POMM) for Life Cycle Software Support (LCSS) has never been adequately justified at the DA level, and therefore has continued to be underfunded and cut annually. This situation has developed to the extent that there are insufficient funds to support the software in many tactical systems.



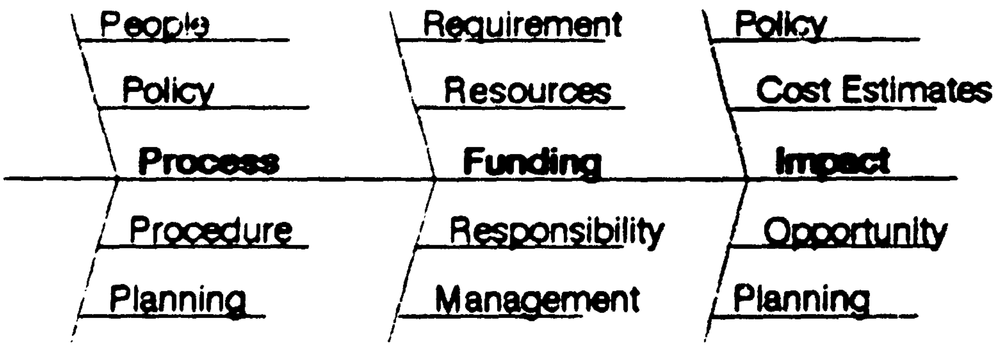
Little Basis for Software Cost Estimates

The intrinsic nature of software and the labor cost necessary to produce quality software place software in its own category. Software is totally different than hardware. It is difficult to show an analogy for the production of software. There is no quantifiable end product that can be seen or felt. The mere fact that software is an inexact science, compared to the exacting, disciplined hardware engineering world, also increases software cost and makes it difficult to define its worth or value. This is what an engineer is faced with when trying to estimate the cost of a software development. Rarely are cost estimate properly conducted by the Army, and when they are accuracy is generally less than 50%. This leads to the perception that it is not possible to conduct an accurate cost estimate.



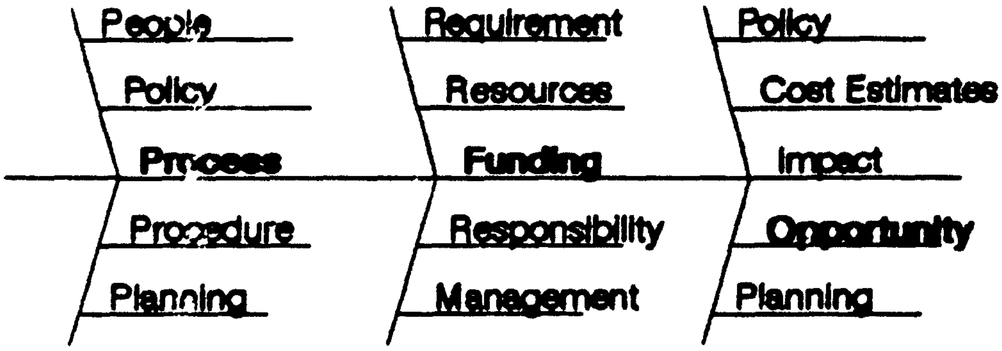
Inability to Show Impact of Underfunding

There are numerous models that aid in predicting the costs associated with the LCSECS, but the knowledge base to define the requirements data is also an inexacting science. Definition of requirements continually fails to adeq'ately establish a recognized process with a stable foundation. The "how to" for defining requirements always reverts to the "from what", and it is the "from what" that is usually nonexistent. From this very shaky means, the requirements for dollars are built into a program for each system. Invariably, the total cost of the entire program exceeds the "expected cost", and underfunding results. When this happens there is panic because there is no realistic means to show impact based on the original ill-defined requirements.



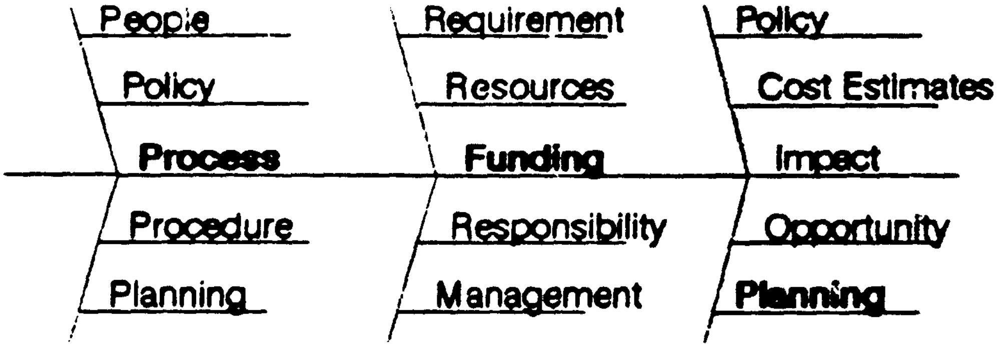
Lost Software Engineering Opportunities

What is the impact of reduced funding or poor fiscal directives? LCSECs are consistently underfunded in both core and system specific dollars, and the result is lost software engineering opportunities. When critical software refinements and enhancements are not accomplished for fielded systems, combat effectiveness diminishes. Technical expertise is lost through the turnover of experienced people. A more insidious effect is the diversion of funds which were intended to support research and development activities in order to meet basic maintenance or sustainment requirements. Without the investment to improve the process and to ensure that new systems are better engineered than existing systems, the Army is just building up larger unfunded liabilities. The opportunity to reduce the support cost of future systems is lost.



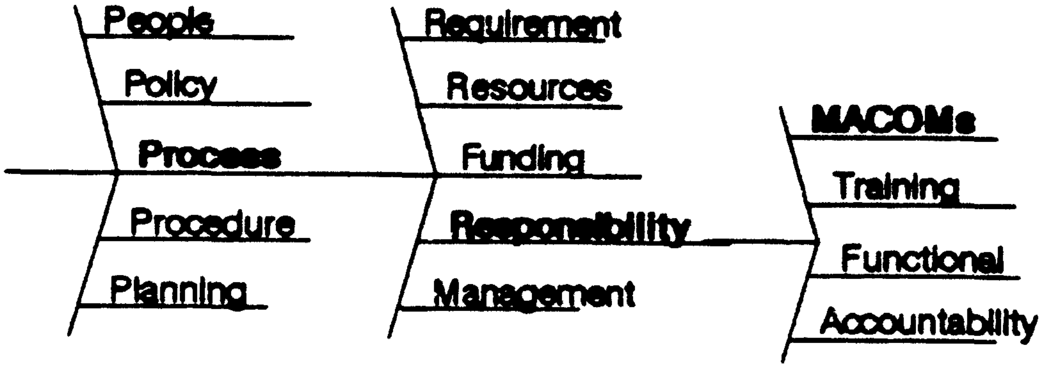
Failure to Execute Realistic Software Planning

Funding policy for LCSECs is so complex that the difficulty in developing a yearly fiscal program has created an undefinable, nonquantifiable monster that generally is interpreted differently at each successively higher level. The LCSEC has a hard time supporting the funding requirements, and each successive level understands the requirements less (or not all), until they reach DA, which has only a "two liner" to justify an exorbitant funding level. As a result, the amount funded has been less than half of the stated requirement. If there were a proponent at each level that could articulate the LCSS story, the program would probably survive; however, this is not the case. This is further confused by those that dictate funding policy, without having the experience or understanding of the total LCSS process.



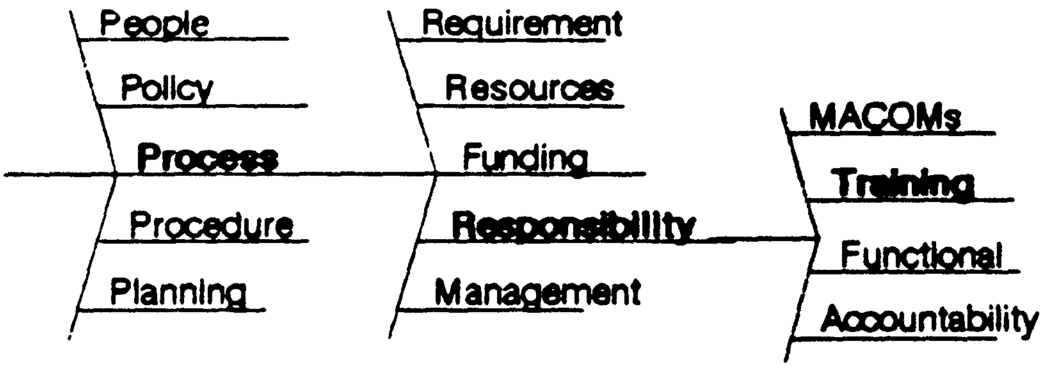
Need Better Handoff Between AMC and TRADOC

Relationships and responsibilities between the AMC LCSECs and the TRADOC, Tactical Software Division (TSD) in five of the six major battlefield functional areas create a well defined and highly cooperative environment. However, the remaining functional area, communications, is a loosely coupled relationship. Factors that contribute to this less than satisfactory situation are no collocation and a combat developer with insufficient experienced personnel to adequately perform their mission. Communications between the PM and TSSM in many systems are also lacking. This has resulted in software modifications being made to a system by the PM without a valid requirement from the TSD. In the area of duplication of effort between the LCSECs and TSDs there is essentially none, they each support different aspects of the development and sustainment process.



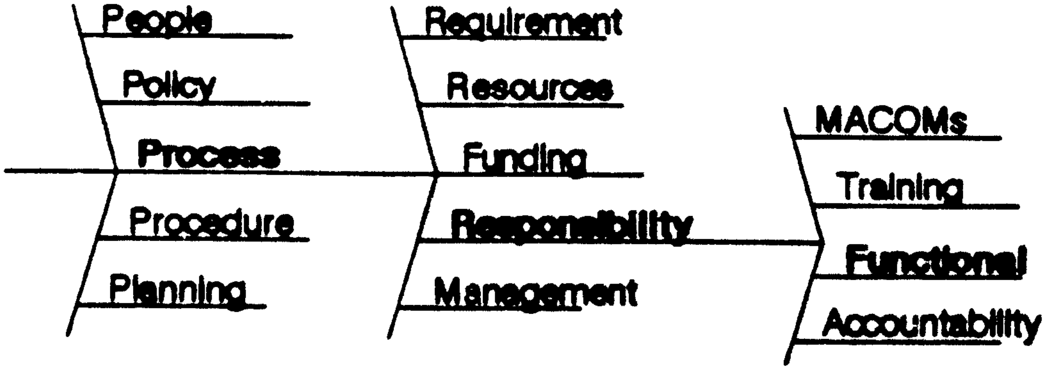
Support for Training Devices has been Fumbled

There is no policy concerning LCSS for training devices. When the PDSS Implementation Plan was revised to become the Life Cycle Software Engineering (LCSE) Plan, training devices were omitted. Funding policy for training devices concerning LCSS is also nonexistent. This has resulted in nonavailability of dollars that has been the underlying reason for AMC LCSECs not accepting training devices that fall within their functional area. The complexity and uniqueness of training device software has been a deterrent with LCSECs wanting to accept something that would require concentrated training to provide the requisite technical support. Another underlying cause is that the lack of policy has nt required the LCSEC's to become involved early in each program initiation.



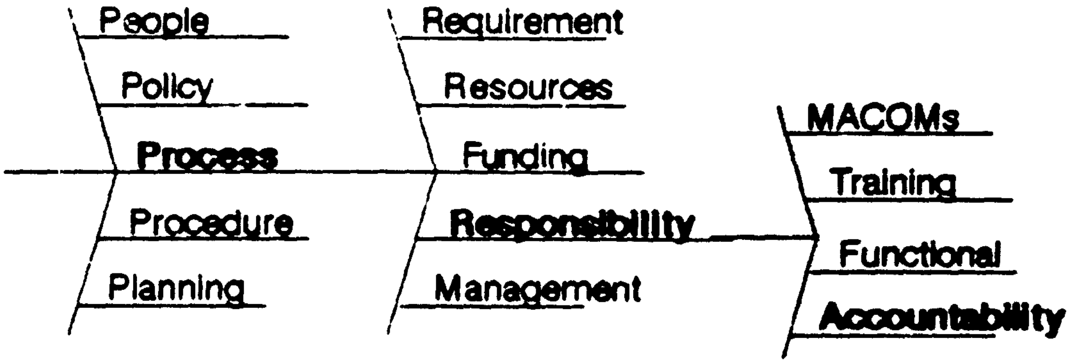
Functional Duplication Still Prevalent

There has been great concern over functional duplication between LCSECs and support activities such as Product Assurance and Test (PA&T) Directorates at each MSC. The only factual documented duplication of effort that exists is between the LCSEC and PA&T. PA&T has the charter to sign (verify) for the MSC Commander on all material releases. In order to perform this task some MSCs have required PA&T to conduct IV&V on each new software version prior to its release and distribution to the field. The LCSEC, however, is charged with ensuring software quality for all software products. It also must certify that every software release is supportable by signing a statement of supportability. Most importantly, the LCSEC must be able to support the software once it is fielded. For the LCSEC to certify supportability and then actually perform modifications to software once it is fielded requires constant training and familiarization with the development. This is accomplished through review of all software deliverables, design and code walk throughs (audits), verification/validation of module and integration, and complete acceptance testing. The PA&T, in the conduct of its IV&V activities, must maintain competence in the same areas and duplicate many of the same activities.



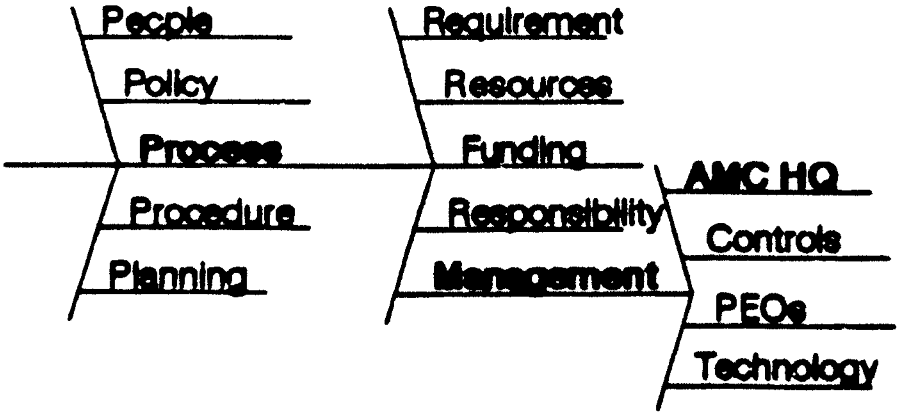
Lack of LCSEC Accountability

There is an Army implementation plan for the establishment of post deployment software support centers. This plan was later revised to incorporate the software life cycle concept, and an LCSS implementation plan was established. These plans detail what the mission and functions of a center are and define the functional areas for each center's system support. Absent from the plan is the requirement for each center to establish internal controls with a mechanism of accountability to AMC. When the LCSS plan was established, systems were defined by functional areas. The plan has not been updated since 1983; therefore, a current database of systems and computer resources is not maintained at each center.



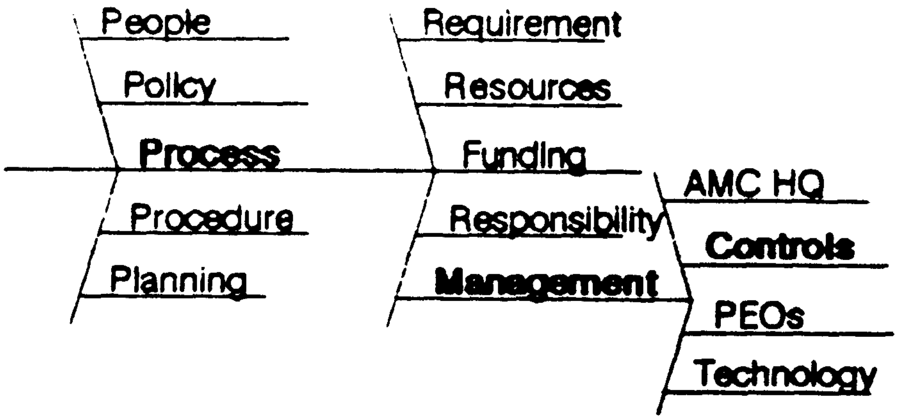
AMC not Effective Player in Software Management

The Army reorganization under the Packard Recommendations established an Army Acquisition Executive (AAE). This reorganization eliminated the Deputy Chief of Staff for Research, Developments and Acquisition, and established the Assistant Secretary of the Army for Research, Development and Acquisition (ASARDA). At the same time it redesignated the Assistant Chief of Staff for Information Management to be the DISC^{4}. Responsibility for weapon systems (software) was transferred from ASARDA to DISC^{4}. DISC^{4} is now taking the philosophy that embedded software is no different than MIS software and therefore is applicable to one Army Regulation, specifically AR-25-series. The reorganization also established the Program Executive Office (PEO) Concept for management of system acquisition. The PEO concept removed PMs from under the technical/fiscal control of AMC and "stove piped" them directly to DA (AAE or DISC^{4}). These two actions have essentially defused AMC's ability to manage software development, but they leave AMC with the responsibility to maintain software. While there was a direct cc rdination link between DCSRDA and AMC, it is not true for DISC4 and the AMC weapons systems staff. Additionally, the AMC weapon system staff was reduced to a level of ineffectiveness concurrently with the DA reorganization. The reorganization has also failed to establish a strong proponent (advocate) for software or a champion in fiscal policy and budget in LCSE matters.



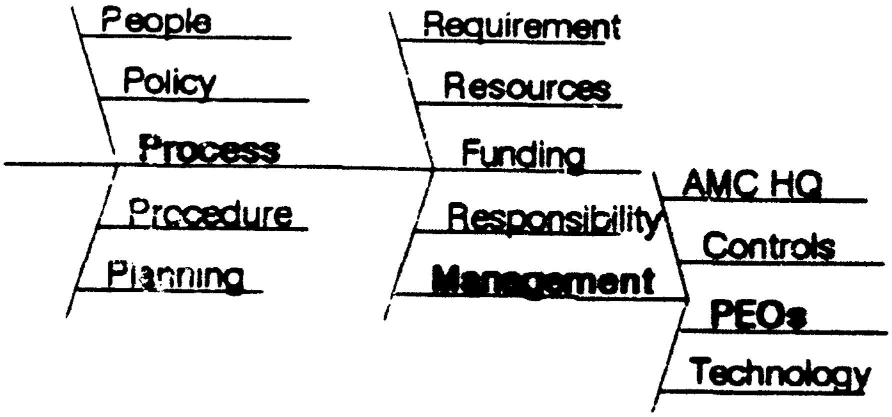
Inadequate Internal Controls

Internal controls for software management are not adequate. The lack or effectiveness of these results in poor planning for embedded systems by insufficient software supportability, inadequate software documentation and software support environments, and nonstandard programming languages.



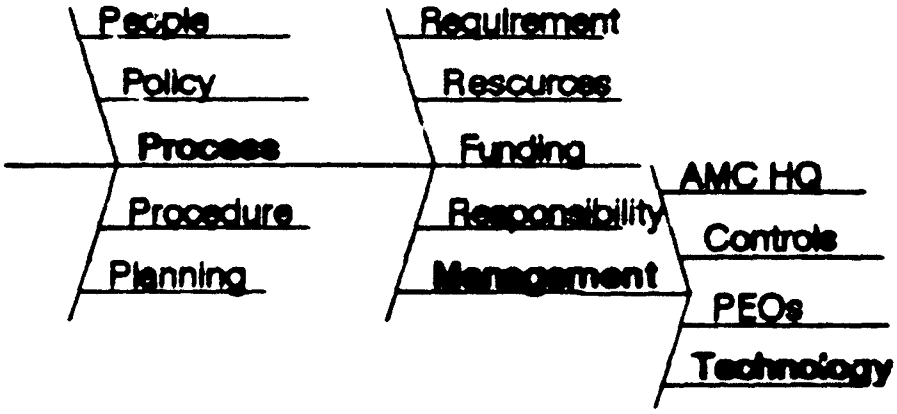
PEOs Appear Independent of AMC Control

The PEO/PM process appears to be operating independently independently of AMC management; therefore, AMC regulations such as 70-16, "Computer Resources Management," have been considered inapplicable by the PEOs/PMs. Another negative aspect is that PMs now have the latitude to look upon an LCSEC's recommendations on software development and support as advisory. The PM may or may not chose to follow the LCSEC's recommendations. Many times sound technical and programatic recommendations are ignored in an attempt to save time or money in the development process.



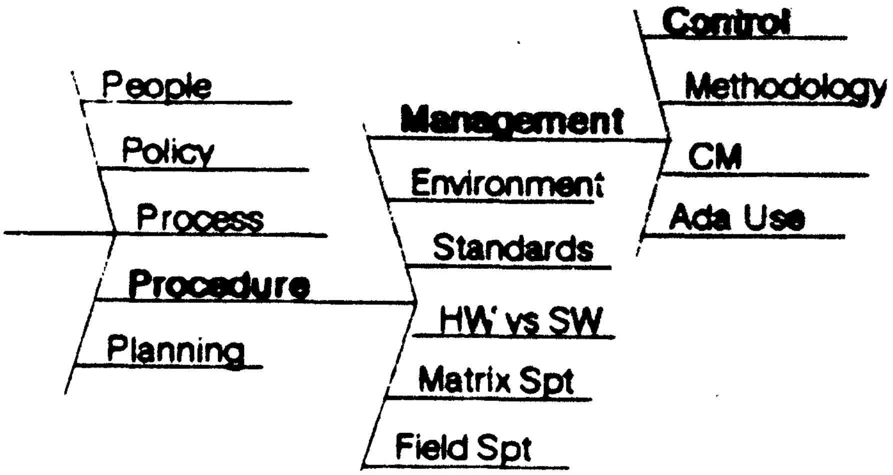
No Army Software Technology Proponent

Technology proponency has been a void at the DA and MACOM levels. This lack of technical direction is probably the most serious factor that undermines the future of the embedded tactical systems program. While some attempts have been made to institute technical advocacy, they have been feeble at best. One such attempt was the establishment of a Software Technology Center to support all AMC Mission Critical Computer Systems. Lacking the technology proponent to provide necessary guidance and resources, the Center has been underfunded and inadequately staffed. This has resulted in a limited number of technology initiatives and has limited the scope of those efforts which were started.



Poor Overall Management and Control of Software

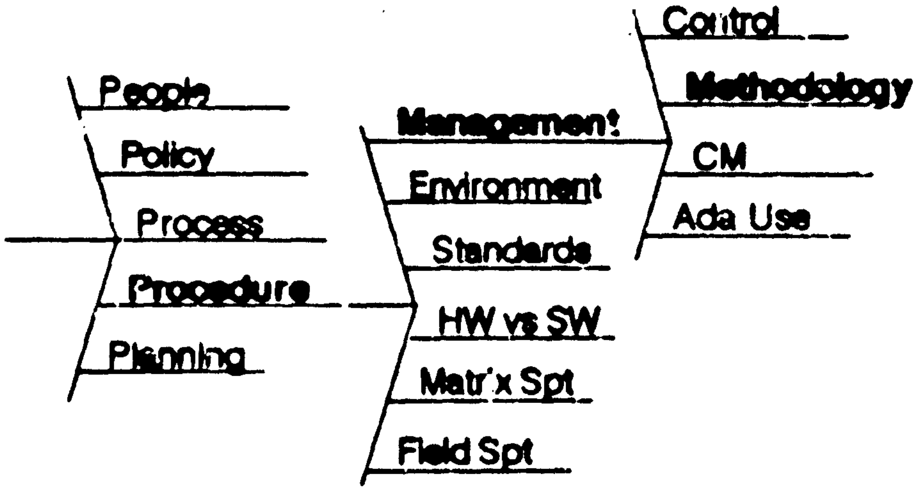
Overall management and control of software engineering practices have been neither acknowledged nor applied sufficiently at LCSECs. This is the result of the splintering of hardware and software life cycle activities without providing adequate authority and communication to integrate a systems approach into software development. The past relationship of the PMs with software and hardware support centers has been based on system-specific agreements instead of sound software management and control practices. Failure to standardize methods in software development inevitably lead to adverse impacts on performance, cost, schedule, and supportability.



Little Management of Development and Methodology

The Army has never instituted a comprehensive process for managing the development of software for MCCR. This void coupled with the lack of a defined methodology has been a key factor in the spiraling of development costs combined with poor quality application software.

The simple fact that software or computer programming is an inexating science begs for discipline. This has introduced the technology of software engineering as the requisite science for instilling the needed discipline; however, this has not been an easy process. AMC and TRADOC have a set policy (means) for developing any new tactical system. This policy has not included the significance of software in cost and as a definable process. Combined with this process has also been the lack of management directives that could be translated into standards to establish the productivity and quality measures necessary for good software.



Configuration Control not Enforced

Software Configuration Control practices throughout AMC are uncoordinated and poorly defined. Issues of concern include:

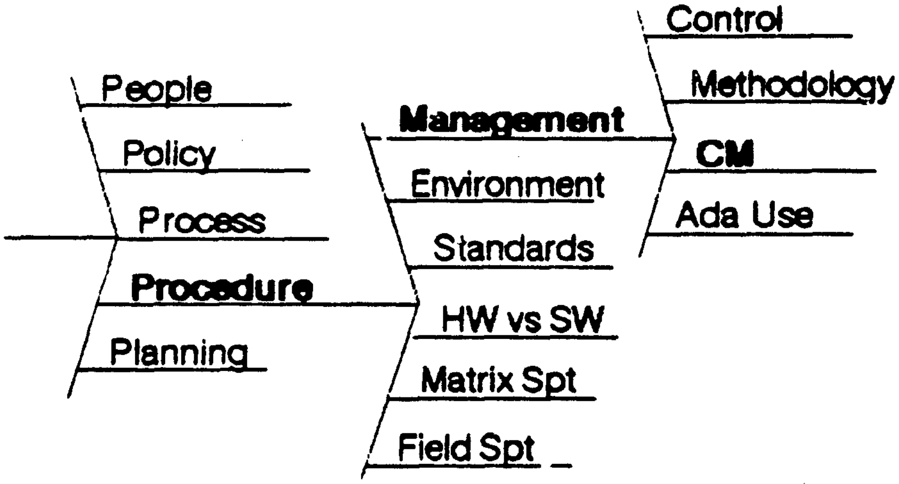
(1) Timely availability of software changes

(2) Adherence to the baseline

(3) Adequate auditing procedures

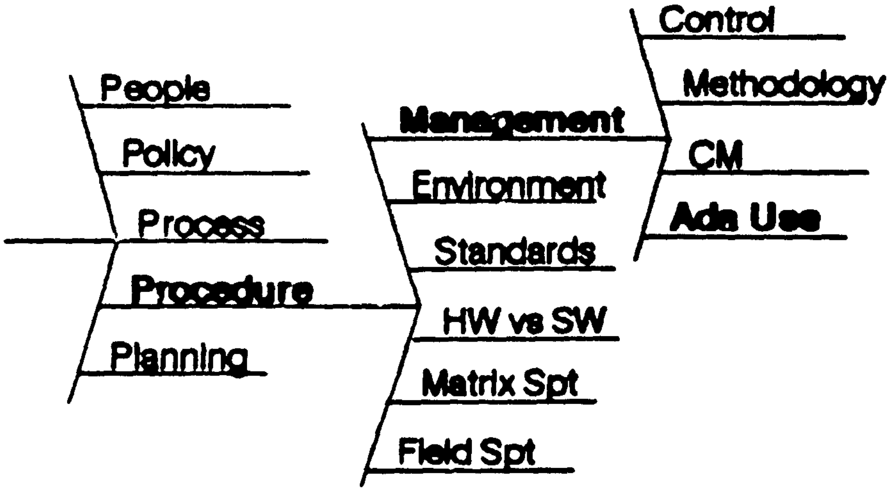
(4) Compatibility of standards and documentation.

System configuration control should ensure that integrated procedures address the total system requirements, including such items as hardware, related CSCIs, support and training elements and facilities, and Government furnished hardware or software, as applicable. CM should also be performed on all non-deliverable software used in the development and on revisions to commercially available computer resources, as described in either the SCMP, SDP, or system CM plan.



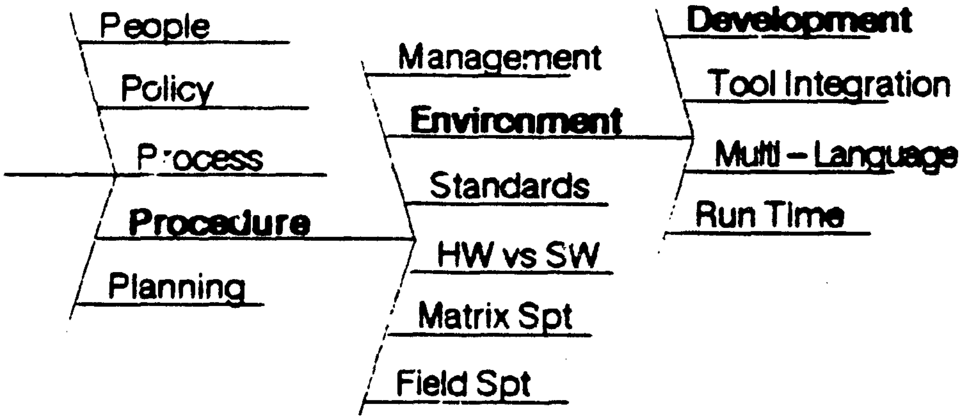
Transition to the Use of Ada Unplanned

The latest Army Ada Introduction Plan is over five years old. In addition, it has only been recently that any Army policy on the use of Ada was forthcoming. AMC policy has specifically established Ada as the PDL and implementation language for major modifications to MCDS software and all new software developments. It should be noted, however, that there is a lack of a formal Army Regulation (AR 70-XX) to state that Ada is the required PDL. PMs have not been indoctrinated with the benefits of utilizing Ada and its programming support environment to the degree that they would make a willing commitment. PMs do not fully realize the necessity to implement Ada in current system acquisitions. The LCSS Implementation Plan has not been updated since 1983 and therefore does not not reflect the current situation with respect to Ada methodologies, tools, or training. New systems that have started with Ada and systems that have been redeveloped (converted) to Ada have experienced painful, costly problems. This can be traced to the lack of a plan for standard Ada tools and training of software engineers and program/project personnel.



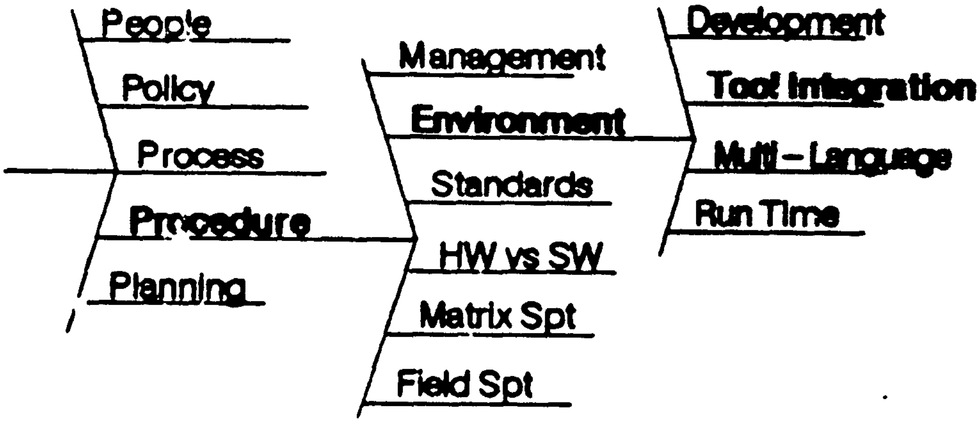
No Strategy for Software Environment Selection

For most systems, there have been no controls placed upon the software development environment. AMC started developing a standard Ada based software environment in 1982. This process can be portrayed as a coat of many colors: Each passing year has taken on a new shade to comply with the current thoughts, with no detailed planning. This evolution ended when the Ada environment effort was cancelled in 1986 with the intent to define commercial tools that would serve as the foundation for the first standard environment. As of 1989 this effort has never been initiated or funded.



Software tools not Integrated; lack solid Foundation

While the selection and engineering of computer hardware for a software support environment has been relatively uncomplicated, the emergence of software development and support tools continues to pose difficult technical challenges. The absence of any effective standard languages, host computers, and target processors has compounded the problem because the embedded support environments are target/language specific. Even discounting these factors, however, the Army faces a significant problem because the tools which have been used to build our systems have no foundation in any sound methological basis nor are they integrated in any meaningful way. The need for the definition of a technologically sound, complete integrated complement of tools must be the basis for technological productivity of software development.



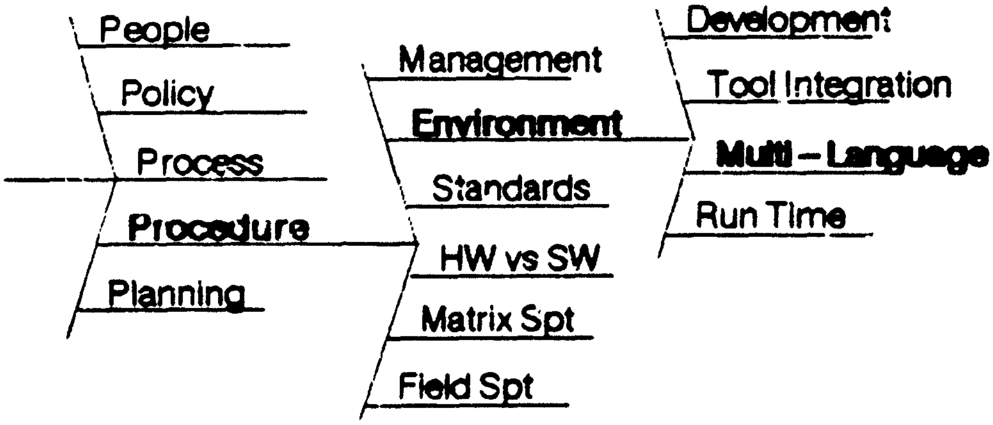
People
Policy
Process
Procedure
Planning

Management
Environment
Standards
HW vs SW
Matrix Spt
Field Spt

Development
Tool Integration
Multi-Language
Run Time

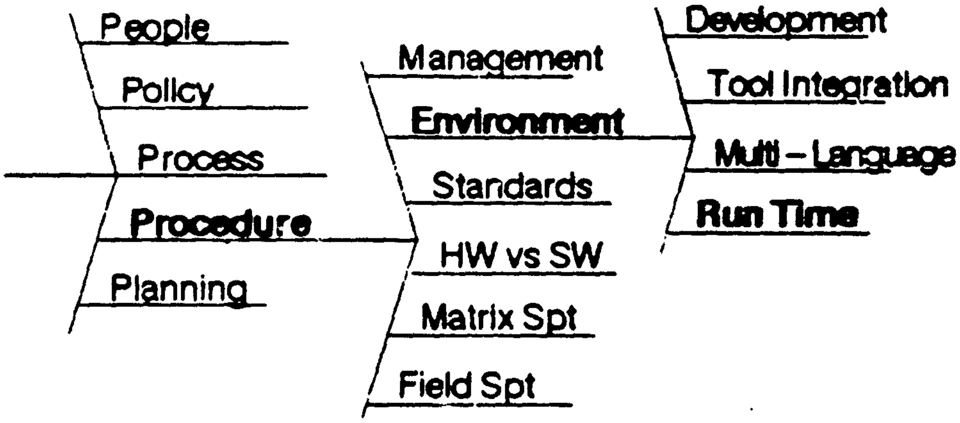
Army still must cope with Numerous Languages

The Army still must cope with numerous languages. This has occurred through the lack of standardization in computer processors and languages. With a proliferation of target microprocessors there has been a high requiremert for support environments to host the uncommon targets. This also is true with the proliferation of computer languages. The requisite tool set for a support environment tha' hosts three different languages, as compared to one, is far greater. MCCR presently total 143 languages and 92 microprocessors that are all different through customized chips and programming languages. What does this mean in terms of resources to support? Cost is skewed toward training software engineers to comprehend more than one language and hardware system, in addition to the need to use very costly environments of host/target computers for each unique language and associated processor. There must then be a maintenance support environment to support such uniqueness with the requisite strategy to upgrade each system during its lifecycle.



Uncontrolled Non – Standard Run – Time Environments

The demise of the military computer family eventually led to the creation of the Army Command and Control Common Hardware Software (CHS) System. It is the PEO CCS charter to lay the architecture for a common hardware/software for each of the functional nodes within the star for battlefield command and control interoperability. It is important to note that this architecture provides the standards for Army CHS; therefore, the entire area outside of the nodal integration has no direction or mandate concerning standardization of computer systems. Thus, the creation of common environments, targets/hosts and runtime environments is non-existent for most embedded computer systems.



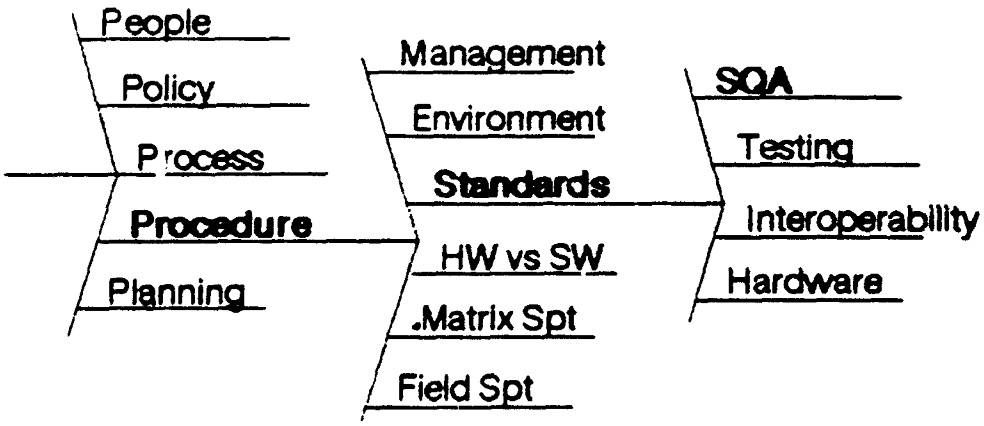
Inconsistent Approach to SQA

Software Quality Assurance (SQA) responsibilities, procedures, and authority within MSCs require some updating and more complete implementation. Issues of concern include:

(1) Ensuring that transitioned software and documentation are of good quality

(2) Conducting quality inspections of computer program code and documentation to ensure supportability of the delivered software product

(3) Determining what life cycle phases PA&T should support.



Testing Tools, Measurable Standards Undefined

LCSEC test and validation processes lack the comprehensive and disciplined practices, centrally coordinated, to meet mission critical requirements. Issues of concern include:

(1) Developing design techniques and methodology for selecting/ developing automated test tools

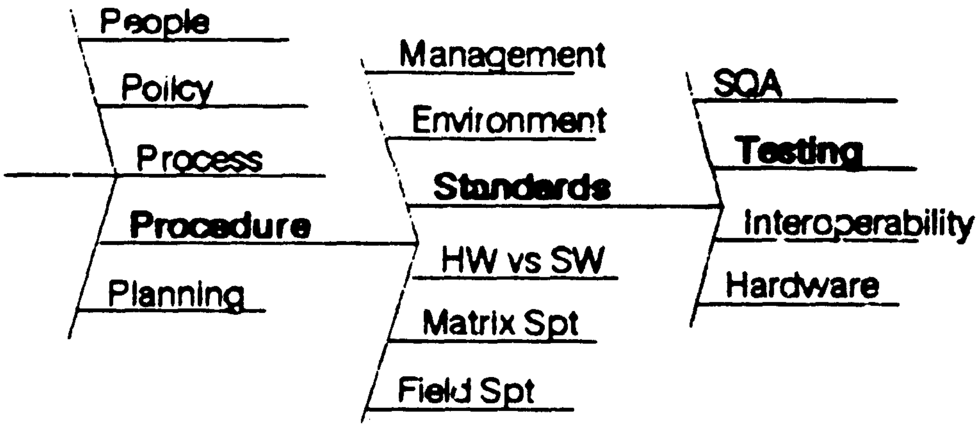
(2) Using simulators and stimulators to a greater extent

(3) Adequately reviewing procedures and results

(4) Creating integration methodologies that include rigorous planning and test criteria

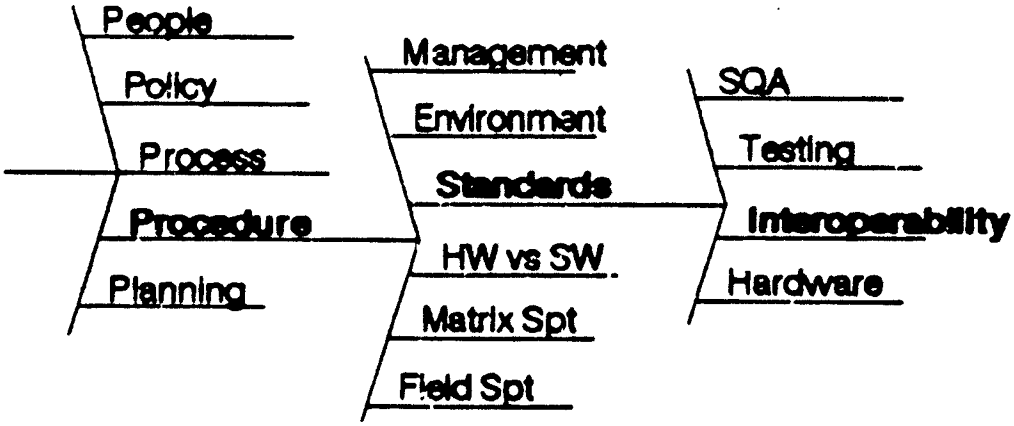
(5) Ensuring responsible organizational levels of testing

(6) Controlling a proliferation of test tools.



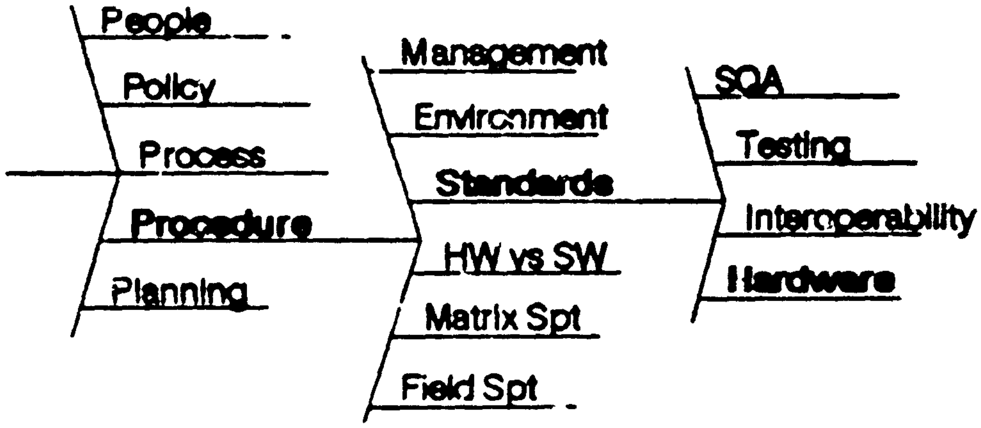
Non – Standard Interfaces; Lack of Test Bed

A need exists to establish Army-wide responsibility and authority to assure that proposed system changes do not adversely impact the interoperability of existing systems. If system interoperability is a specified system requirement, then management of software must provide a mechanism for determining whether any proposed changes will disrupt interoperations. Further, a mechanism must exist for analyzing the impact of imposing new interoperability requirements on existing systems. In order to maintain interoperability among fielded systems, installation planning for MCDS software revisions must be coordinated. This coordination must account for revisions affecting inter-site compatibility and MCDSs that must be interoperable with other and different MCDSs. Policy needs to be established for developing new tactical data links and for maintaining existing data links. The Army has allowed development of too many uniquely defined data links with very similar overall requirements (ATDL-1, MBDL, Patriot DL, FAAD C21 DL).



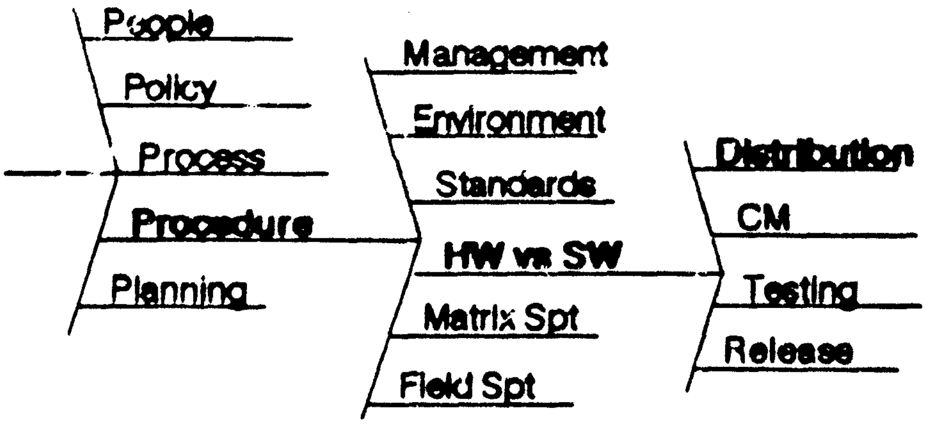
Target Hardware, Media Variety Complicates Support

As a result of the way the Army let its contractors select the hardware which makes up its fielded weapon systems, it has inherited two problems which add additional unnecessary complications to its software support problems. The wide variety of target processors have made it virtually impossible to establish a common run time support for its weapon systems. For each of its field systsms, it not only must maintain the operational software, but must maintain the elements of the field operating system on which the operational software runs. The lack of standard I/O devices and media require the continues acquisition and support of a variety of equipment just to provide software fixes and replacements to the field.



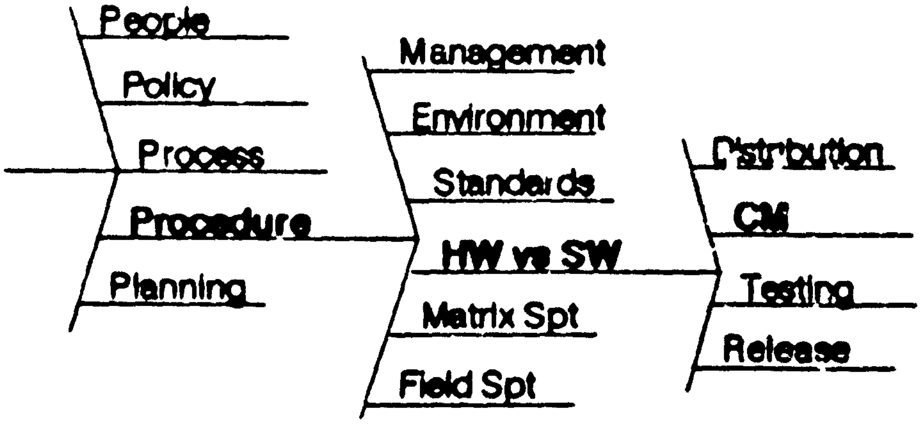
Poor Execution of Replication, Duplication & Control

Because each system is being developed individually under separate PM control, with each system having established its own unique method of software replication and distribution, several problems result. Current practices and procedures in this area make inefficient use of scarce personnel resources, do not allow for timely replacement of failed software/firmware (SW/FW), and do not lend themselves to a timely upgrade of systems in fielded units. Because there is no common/central configuration control point for the fielded SW/FW, the probability of field units receiving incorrect software or firmware is increased. The lack of central control and a standardized methodology increase the problems of accountability of stock/replacement of software/firmware components. Because of these factors and the lack of a prioritization scheme based upon user needs, timely replacement of failed media is difficult or impossible. Some systems have duplicate software media; others have the ability to replicate their software in the field, still others can be replicated only by the development contractor at the development site.



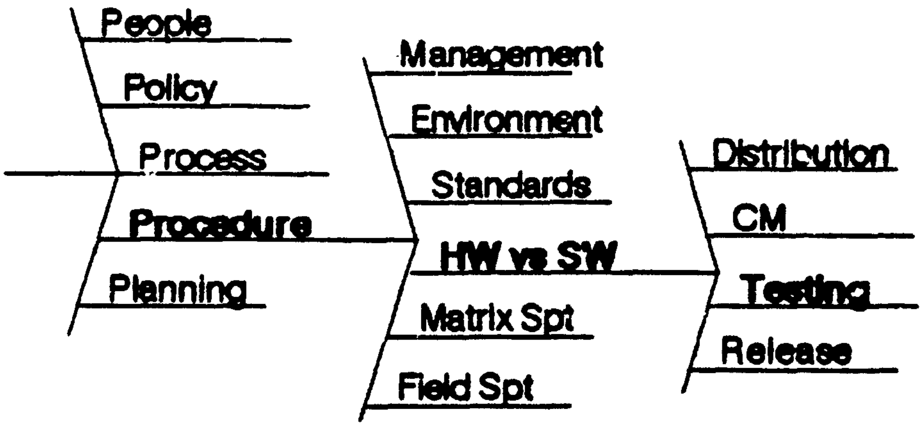
No Effective Configuration Management for Software

The Army is faced with managing rapidly changing software technologies and a multiplicity of complex hardware and software systems in the field. The ever-increasing costs of new software development and existing system life cycle support are overstressing resources and could threaten the combat readiness of the US Army. Existing CM procedures are highly coupled to hardware system requirements. These directives and standards, which preceded new Joint Logistic Commanders' software policies, are not fully responsive to the dynamic needs of MCDS software development, acquisition, and life cycle support. As a result, there is little commonality in the way different MSCs implement software configuration control. AMC (AMCDE-SB-C) on 2 December 1986 issued Commander's Guidance Statement (CGS) No. 155 for Mission Critical System Software, Battlefield Automated Software Development. This guidance, which implemented DoD-STD-2167 and DoD-STD-1467, was incomplete with respect to software configuration management standards required to support software life cycle support. New guidance specifically tailored to the unique software management requirements is a pressing need.



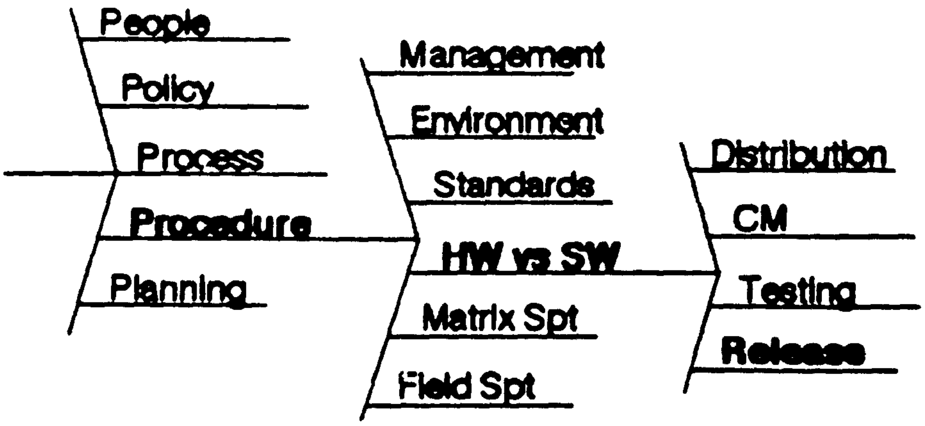
Software "Components" not Adequately Tested

DoDD 5000.3 (Test and Evaluaticn) is the authority for test and evaluation in the acquisition of defense systems and also provides definitions and guidelines for the Test and Evaluation Master Plan (TEMP). The Army's implementation of this DoDD is AR 70-10 dated 30 April 1986. DA published the LCSS Implementation Plan dated Dec 83 to establish a strategy for Army-wide management control and software support of MCDSs. The plan advocates software testing consistent with the procedures specified in the Software Quality Assurance Plan for each specific system. The categories of testing included Certification Testing, Verification and Validation (V&V) Testing, User Acceptance Testing, Development Test (DT)/Operational Test (OT), and Interoperability Test. Although the draft Software Test and Evaluation Manual (DoD 5000.3-M-3) provides a software evaluation guide (Appendix A) and a software test and evaluation plan (Appendix B), it does not present specific guidelines as to a threshold percentage of test pass/fail criteria, distinction between critical requirement function tests versus non-critical requirement function tests and their relative weights in determining the pass/fail criteria, and a minimum set of performance function tests required for acceptance.



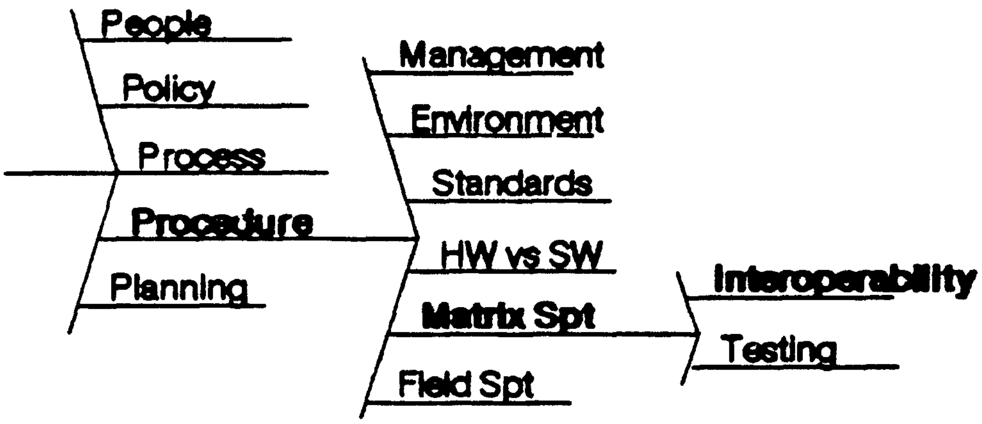
Lack of Management of Software Releases

The release of a software version for any one system is akin to the iceberg analogy: what you dor't see :re the problems associated with inadequate testing, interoperability, documentation for the field user, and evalution of compliance with the use.'s requirements. Changing software in any system is the responsibility of the material developer, but the responsibility of validating the requirements rests with the combat developer. For hardware releases, there is a standard, well-exercised process which ensures that all concerns have been addressed prior to release. Current policies call for this formal material release process to be followed for all "major" software releases. Unfortunately, some MSCs have classified very few of their software changes as major, and therefore they have not been released trough the formal process. The ambiguity in current material release regulations needs to be removed. It should be the rule rather than the exception to review software releases, and the process needs to account ior the peculiarities of software including interoperability certification .



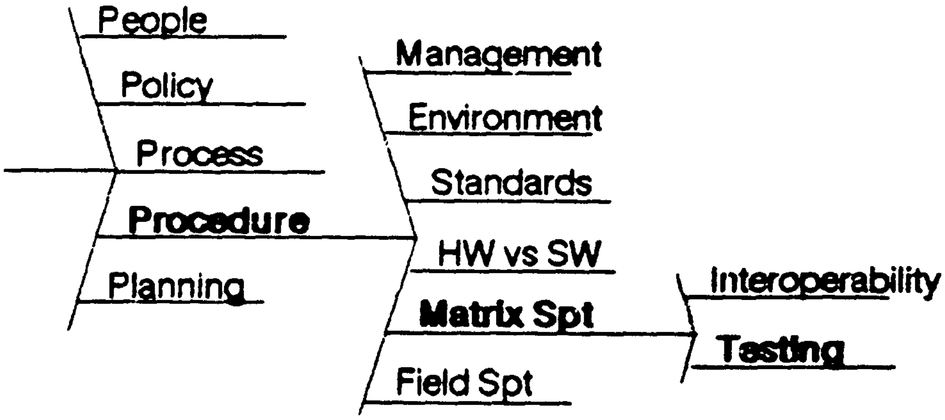
Poor Consideration of Interoperability

While general testing principles apply, interoperability testing is unique from other forms of testing in that two or more systems are required. While certain aspects of interoperability may be tested in a stand-alone system (e.g., ability to send and receive messages using specified protocols), true interoperability testing is complex and necessitates a special approach in formulating test requirements and test capabilities. Use of an interoperability test capability is required to validate system interfaces and interoperability requirements. This is a major issue that encompasses pre- and post-fielding. Prior to fielding, major systems must be tested to assure that they will interoperate properly in the field. This methodology has not been implemented and there are serious concerns about whether it can be implemented properly in the near future without an integrated interoperability testing capability as more new systems are fielded and interoperability requirements become more complex.



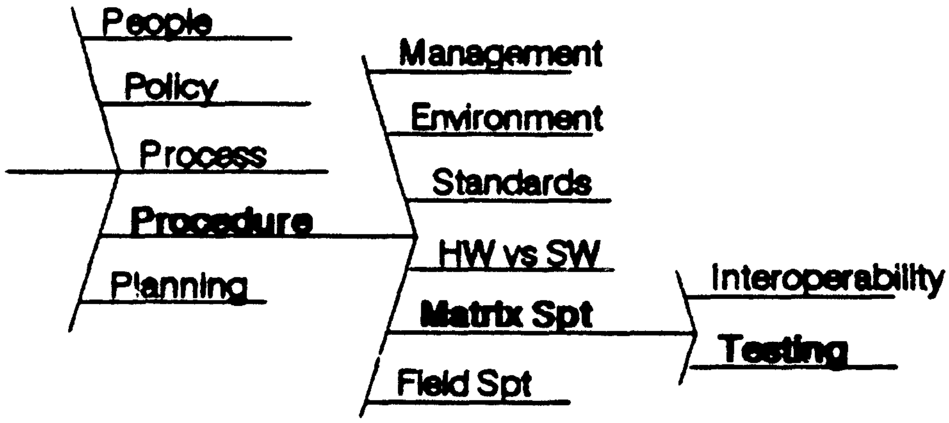
Lack of Interoperability Test Bed

After fielding, the system users need to be able to discern the cause or basis of a system failure. The question of which system is at fault is not a trivial one to answer. The system engineering talent is not available in the field, and it is not clear to the user as to whom should be called. An interoperability test bed would provide the required capability to validate interfaces, resolve cross system difficulties, and determine which agency is responsible for correcting interface problems.



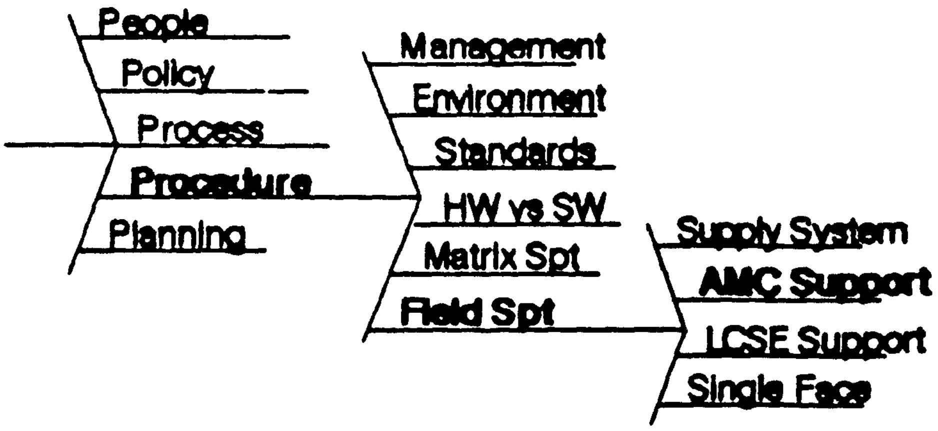
Lack of Interoperability Test Bed

A great value inherent with software today is the ease of making a rapid change (enhancement) to a system in response to a requirement change(threat). For this to happen a support system must ensure the following: interoperability consideration and a means to react in a timely manner. Both of these requirements are difficult to meet if standard Army systems are used. The interoperability problem has already been discussed; timeliness has not. Pushing fixes through the hardware supply system and not effectively be used to get emergency fixes to the field. When interoperability is needed, the supply system will not work at all when interoperating systems must implement new software simultaneously. This thwarts the whole benefit of the software's ability to effect a quick fix.



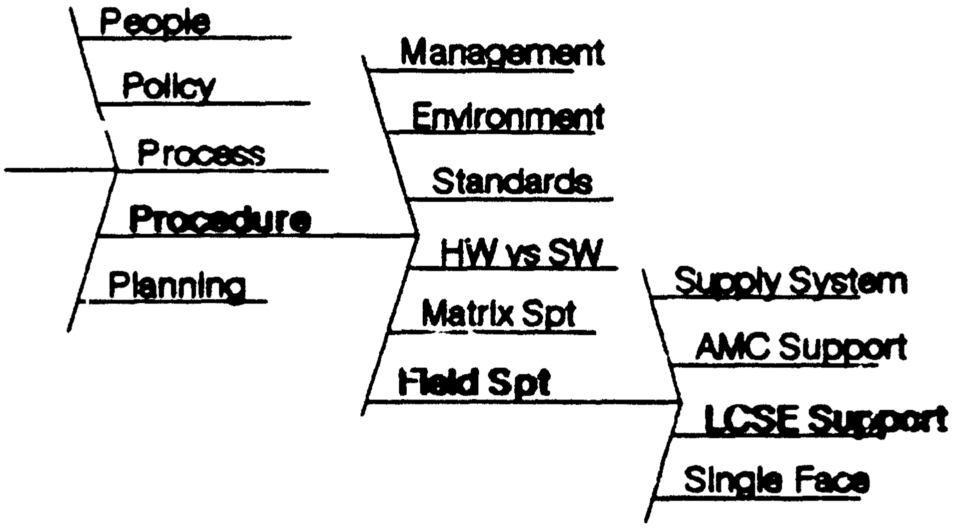
Untimely Response to Field Problems

Although there is a Quality Deficiency Report/Equipment Incident Report (QDR/EIR) Reporting System, it has not been modified to incorporate software deficiencies. This, coupled with the ever present problem of identifying software vs. hardware troubles, can result in delays in fixing problems. There is no easy way to track computer deficiencies through the QDR/EIR system. DA PAM 738-750 does not have a code that describes a problem associated with computer hardware/software. The embedded computer mission critical defense systems deployed or near deployment to Army field users employ numerous types and variations of media manufactured to load operational software into the individual system. To date no effective effort has been applied by AMC to standardize on a limited set of such media. The result is that a unique production tape replication facility is required for virtually every system. For example, one cartridge utilized for five Army systems has a unit cost of $500-600, yet inspection of the cartridge indicates little basic technological difference with respect to VHS tape cassettes, which sell for one percent of the cost. The difference is volume. The Army is procuring their own cartridges in quantities of thousands per year while VHS cartridges are being manufactured and purchased in quantities of millions per year.



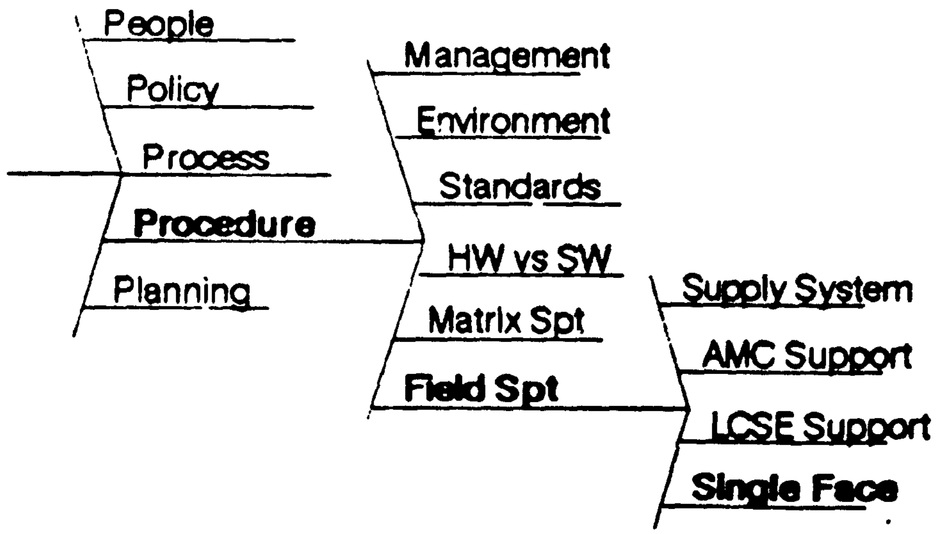
Uneven Usage of Area Software Analysts

Operators and maintenance personnel are not trained to detect and verify software problems within an embedded system. MSCs attempted to fix this problem with the creation of Area Software Analysts. The ASA does not work under the direction of the LCSECs, but is assigned to the Readiness side of each MSC as are the LARs. There is no process or training for the ASAs to familiarize themselves with the embedded systems they must support. Their best effort is to "possibly" identify a software problem and report it to the appropriate LCSEC, although in most instances the user, if knowledgeable of the LCSEC, will report the problem directly.



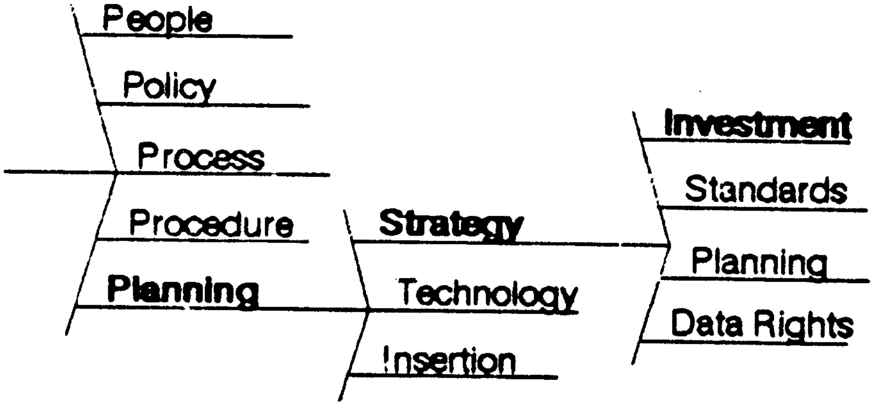
PMs, LCSECs Each Need to Deal Directly with User

Since the first BAS was introduced to the field there has never been an approved, funded plan to establish AMC forward LCSEC offices in OCONUS. The result of this has dictated the requirement for a new software team each time a version is released to the field to ensure successful installation and operator training. In many cases, there are no standard or common methods for training either the field users or the software field support personnel. The situation is further complicated by the replication and distribution methods being as diverse as the systems themselves. Some efforts by organizations such as PA&T, LCSECs, and DRE, appear to be redundant, which causes inefficient utilization of an already critically small pool of experienced personnel. The overall system level analysis, evaluation process, and the preparation of the ancillary support packages/functions are difficult to implement in a timely and coordinated manner, which causes additional problems in distribution of complete system packages to the field users. Documentation and/or training, for example, frequently lags the delivery of the actual software media.



Little Investment in Tools, Reuse

A strategy for developing a program that will identify the dollars for investing in standard environments and compatible tools is still a long way off. This has been stymied because there is no focal point at the DA level that has the where-with-all for planning software technology. Evidence of policy or regulations that support definition of a strategy is absent; therefore, software domain needs are not considered. Standards and methodologies for software architecture should include procedures for creating portable and reusable software. Here again, there is no directive or regulation to establish a mechanism/organization that would be responsible for one-time developed and reusable software tools.



Incompatible Tools; No Agreed Upon Methodology

Because there is no agreed upon methodology, there lacks the means to answer many important questions:

(1) How can generic requirements be established?

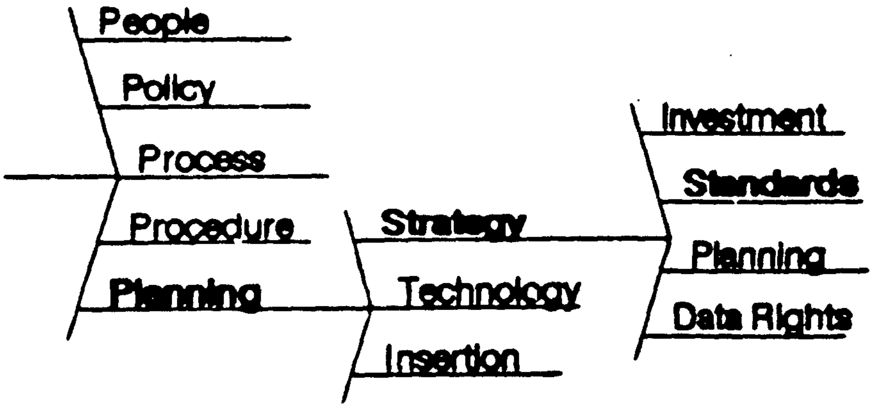
(2) How can one identify and develop/procure common software with reuse as a goal?

(3) What standards apply to test?

(4) What will be the incentives to the contractor for reuse and pass off of tool development?

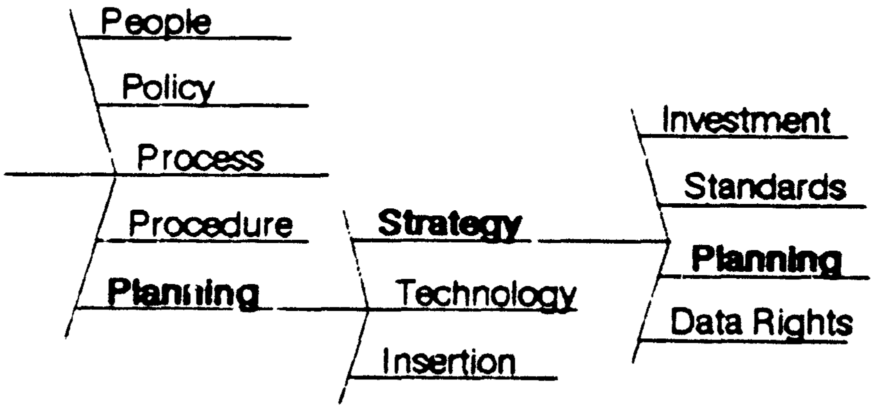
(5) Would there be enough common software identifiable to justify the cost?

There has been little or no consideration given to planning for a software development or support environments so that a suite of compatible, complementary tools were acquired. Rather than acquiring individual packages suitable for their own individual purpose, identifying tools capable of being integrated with others for greater effectiveness should have been the direction taken.



Software Concerns Addressed as an Afterthought

MCCS procurement strategy in the past has involved the prime contractor approach. The contractor develops the software from scratch and may employ proprietary software tools, and in the worst case, use an unapproved HOL. During negotiations, the contractor will resist the applicable military standards and use cost and schedule impact as an argument to reduce the scope of software documentation deliverables and other administrative and/or management-type responsibilities. The result is that the eventual fielded system software is difficult to support. Documentation may be inadequate, often inaccurate, and can be in contractor-unique format. Reusable and transportable software will not have been addressed. Support may eventually require a sole source contract, back to the system prime contractor, and the adverse process continues.



Failure to Address Data Rights, Liability, Warranty

A new approach is required in procurement practices, which may require substantial contract incentives, to encourage positive response to Ada, interoperability, NDI, and commonality in software requirements. To be successful, these contracting practices will have to be coupled with complete compliance to new specifications and standards. Because of some of these difficulties inherent in the acquisition of software, we have failed to provide definitive guidance to address critical questions:

(1) What level of government data rights is required for each individual procurement?

(2) When is the optimal time to acquire data rights, if any?

(3) What is the proprietary right when one contractor develops tools and another contractor maintains?

(4) What derived right does slightly modified software resold by a contractor to another buyer at a fraction of cost have?

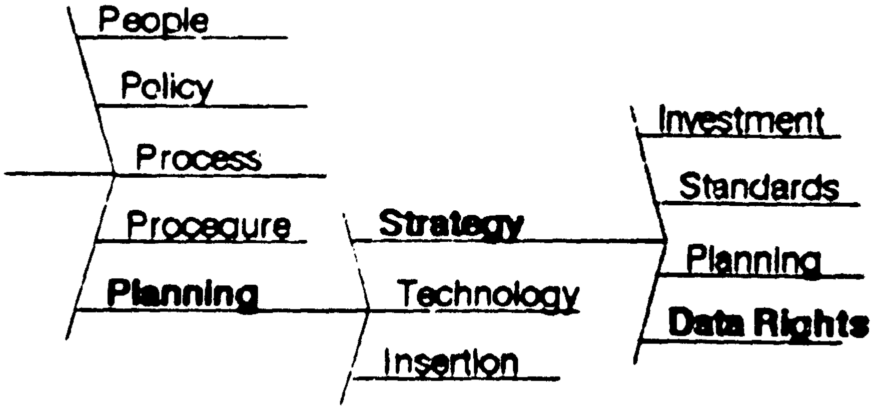
(5) Who is liable for software that contributed to a human death?

(6) For what time frame should warranties be required to ensure that technical deficiencies will be corrected?

(7) What is the original developer's responsibility (warranty) if a second contractor performs enhancements?

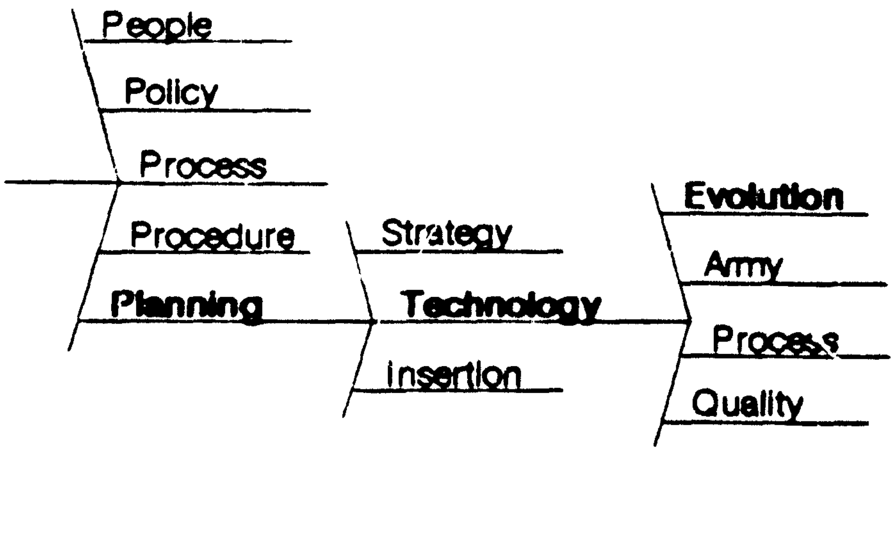
(8) What is covered in a warranty, and how can all implications be anticipated?

(9) Will warranties induce great quality of software?



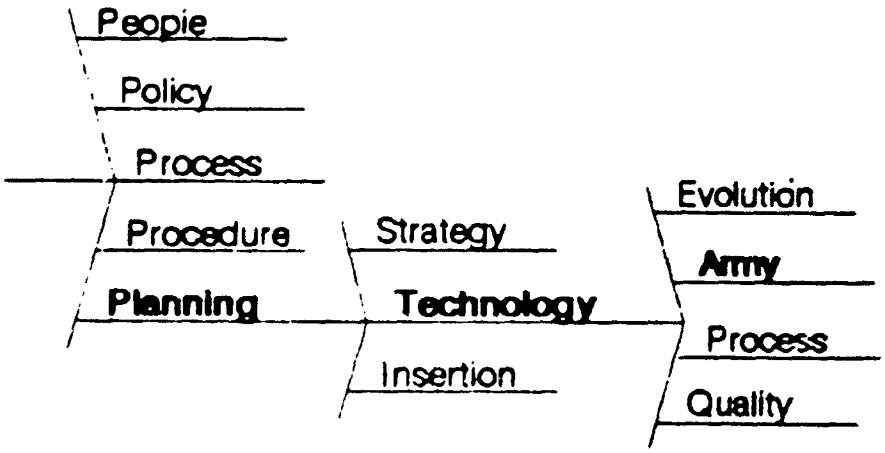
No Software Technology Program

There is much written and taught concerning the discipline of software engineering; however, the actual practice is still in the infantile stage. The evolution of software engineering as a disciplined science has only emerged in the last decade, yet the computer has been in existence for over half a century. Software technology has typically found its way in the unstructured cottage industry, unlike the directed research of electronic (hardware) technology that has been a highly disciplined, standardized science. The nature of this evolution lead professionals to continually reinvent the "software wheel" without taking the time (and resources) to instill a discipline into the science. Another phenomenon of the "wizardry art" of software was the incredible momentum that propelled its technology explosion. While this explosion of software technology enabled man to walk on the moon, it was not harnessed to instill discipline toward a productive end. This rapid change also made it difficult to set standards and policy in place because there simply were no developed skills to begin technical management. However, approach to maturing software technology areas, planning for and implementing advancing technologies will be fragmented, haphazard, and ineffective.



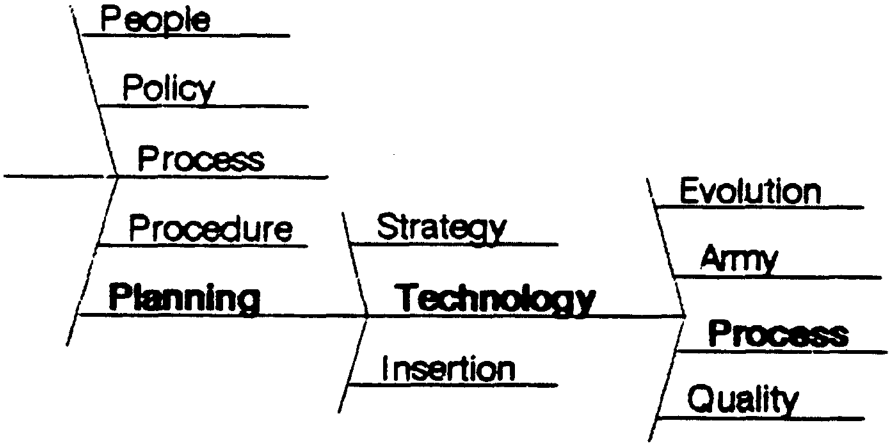
No Concern with High Performance Systems

Although contractors have successfully been building large software systems for some time, quantum leaps in the sophistication and complexity of emerging systems have placed new and more rigid requirements on their development methodologies, design considerations, and implementation processes. In particular, high performance requirements such as increased reliability, real-time operations, and the handling fusion of huge amounts of data have stressed contractors' ability to produce systems capable of functioning for prolonged periods under battlefield conditions. We have further compounded our problems by virtually ignoring interoperability issues during systems' concept and early development phases, consequently fielding systems that perform adequately in a discrete envrionment, but cannot meet the high performance requirements of an integrated battlefield.



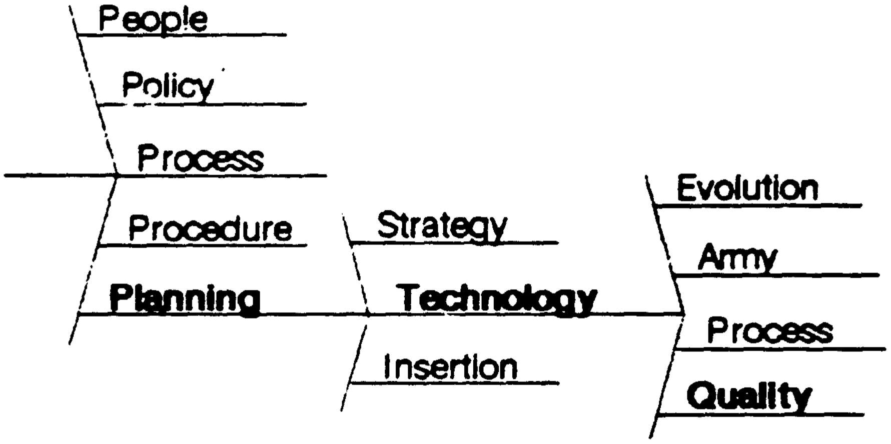
Failure to Bring Discipline to Software Management

Failure to insert discipline into the software management process failed through the lack of standards and a viable methodology that inevitably leads to adverse impacts on performance, cost, schedule, and supportability. This led to such a dynamic fiscal program that only the best software management "wizard" was skilled in defending development and sustainment costs, since there were no realistic criteria for software program planning. Past relationships of the PM's with software support centers has been based on system specific agreements instead of sound software management and control practices. This was all the result of splintering software life cycle activities without providing adequate authority and communication to integrate a systems approach into software development. The management of a system (BAS) has not included the significance of software in cost or quality as a standardize definable process.



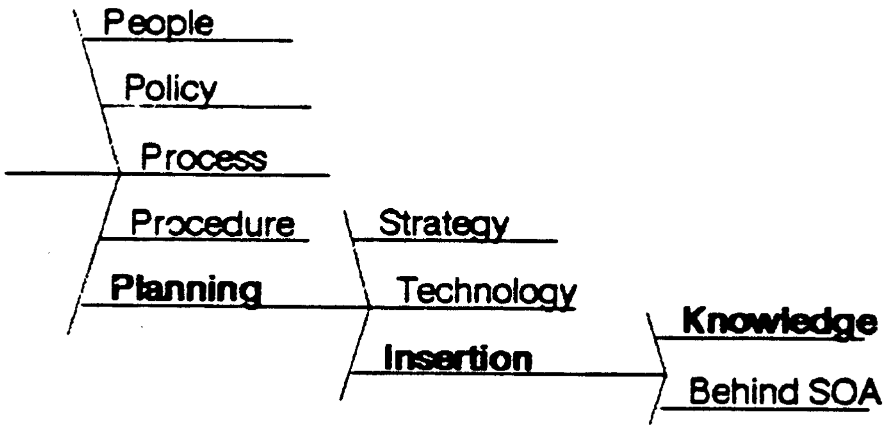
Software Quality Frequently Traded – Off

Systems are developed today, and yet there is no universal measure of software quality. With all the emphasis placed on developing quality software, effectively measuring it is still uncertain. However, what little is known about developing quality software usually is not written into the contract specifications or is degraded because of schedule, cost, or quality tradeoffs. All of this results from the absence of a standard development methodology that desperately needs engineering discipline. Because there is no measure for accessing the quality of software, quality will always be a function of the caliber of the people developing it.



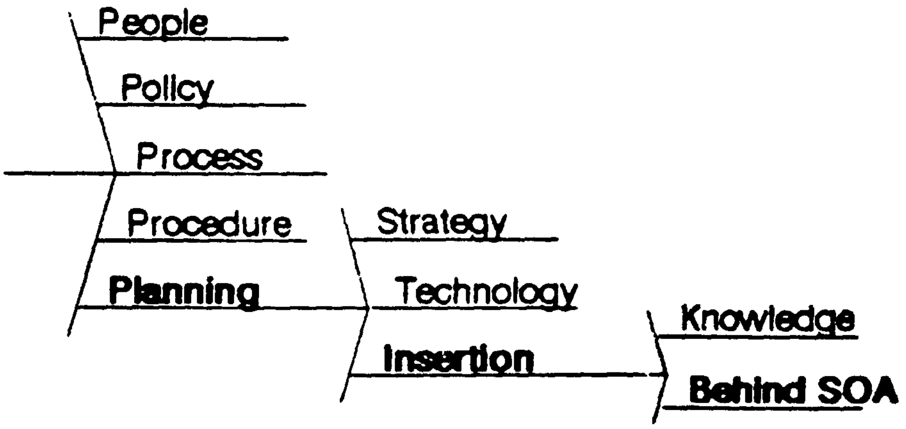
Failure to Maintain Software Knowledge Base

Altnough software is a critical element in our continued ability to develop advanced weapons systems, we have failed to recognize, and to provide sufficient means and incentives, to foster the growth of a software knowledge base in our government staffs. This problem area has been discussed under training, management, awareness, education, and career path issues, and effects both government and civilian and military personnel. Without the expertise to understand software technologies and to effectively oversee program acquisitions, no level of financial investment will return to us the systems capabilities and fighting capabilities winning will require.



Army State – of – Practive Far Behind State – of – the – Art

Why then is technology not inserted into a software discipline to keep state-of-practice in-line with state-of-the-art? Training of software engineers is not geared toward an evolving technology and its growth in software system complexity. Training must orient toward a "systems" software approach and instill the interdiscipline of a software engineering methodology. Training is only one part to solving the puzzle since quality productivity is a key missing piece. Productivity translates to a lack of standard tools and techniques to reduce development and training costs. Right now there is no incentive for developing these tools and maintaining research facilities because the cost is viewed as too great. It is difficult to conduct a trade off analysis of improved productivity versus the cost of developing new tools and better training. If there is not a concentrated effort to place the dollars up front in research and initial development then a far greater cost will result. This adverse cycle has been experienced time and time again, yet only now is the DoD waking up to the high fiscal reality of it all.



APPENDIX B

Recommended Actions

"In the design of automobiles, the knowledge that you can design the motor more or less independently of the wheels is an important insight, an important part of an automobile designer's trade. In our field, if there are a few specific things to be produced ... it would be very important to decide what are their parts and what is the proper sequence of deciding on their parts."

Peter Naur
NATO Science Committee
conference on Software Engineering
October 7 – 11, 1968

SS-111
Implement an Effective Software R&D Strategy

The Army must recognize that, because of the growth of the commercial software market, it needs to be primarily a buyer rather than a builder of technology for software development. The Army strategy for software technology should be built upon the industry standards and use of commercial tools whenever possible. However, in those areas where special Army needs exist, the Army needs to pursue an aggressive, focused R&D program.

RECOMMENDATIONS:

A. Develop a Software Technology Plan

- Identify pacing technology issues
- Define areas where technology crucial to Army needs
- Show areas where industry/government agencies will focus efforts

B. Establish a testbed to evaluate commercial products

C. Evaluate commercial products/standards for Army applicability •

D. Structure R&D program with reuse as a keystone

- Application Software
- Tool selection and use among projects
- Public domain, non-proprietary products

E. Conduct research to satisfy specific unfulfilled Army need

- Software Reuse
- Metrics
- More productive Software Paradigms
- Others to be defined, e.g.
    - – Prototyping & Formal Specification
    - – Requirements/Documentation/Life Cycle Processes
    - – Distributed Computing/Real-Time Performance
    - – Application of Ada Tools & Methodology

F. Structure Incentives to Increase Compatible Industry Software IR&D

SS–112
Develop an Approach to Software Reuse

A consistent approach to software reuse must be developed. Implementing effective software reuse procedures will result in cost savings, improved quality, and reduced development time. The approach must be built on the recognition that it will take at least ten years for reuse technology to mature. Initial efforts will employ reusable software artifacts, follow – on efforts will improve ways to adapt existing systems. In the final phase, reuse will be a function of the tools used to generate new systems.

RECOMMENDATIONS:

A. Develop a plan to acquire reusable software

- Public domain, non – proprietary software
- Based on Industry standards
- License commercially available software

B. Establish a software reuse R&D program

- Build on current constructive approaches
    - Classification and Retrieval Systems
    - Knowledge – based tools
    - Design and coding standards for reuse
    - Creation/maintenance of library of "certified parts"
    - Techniques to make artifacts more general and flexible
- Investigate use of generative systems
    - Near term emphasis on application generators
    - Long term emphasis on template/transformation systems
    - Emphasize approaches using domain/process knowledge

C. Investigate non – technical issues

- Development of standards
- Data rights, warranties, and liability

SS–113
Develop and Evaluate Software Metrics

Process and product oriented metrics need to be defined and evaluated. Tools which support useful metrics should be integrated into software environments used to develop and support Army software. Quantitative measures of contractor performance and product suitability are needed to ensure successful management of the software process.

RECOMMENDATIONS:

A. Collect and use existing compiler performance metrics

B. Establish framework to evaluate metrics for applications

- Identify metrics with high potential for use
- Capitalize on existing effort to identify tools
- Use data from DACS/RADC database where applicable

C. Calibrate existing metrics with ongoing programs

- Process and Management Indicators
- Product Design and Build Attributes
- Performance Indicators

D. Develop and evaluate new product/process metrics

- Map metrics to important decision factors
- Identify immature measures or those based on invalid assumptions
- Evaluate evolving practices and products

SS–114
Evaluate Software Life Cycle Models

Software technology is an immature, rapidly evolving technical field. Dramatic growth in complexity and size of Army software systems requires the Army to foster and direct the evolution of new practices, procedures, and methods for the development of software systems.

RECOMMENDATIONS:

A. Develop model to select paradigm and evaluate payoffs

- Sequential or waterfall model for well – defined requirements
  - – – Needs more intensive up – front planning of interactions
  - – – More comprehensive documentation required
  - – – Will reduce cost if requirements are complete at start
- Rapid Prototype and Iterative Development
  - – – Best to use when user requirements must be refined
  - – – Allows user to provide feedback with use of prototype
- Incremental Development
  - – – Requirements are understood but user needs quick initial fielding
  - – – Functionality and performance are slowly integrated

B. Investigate alternative paradigms

- Automation based
- Reuse Based
- Risk Management based

C. Improve specification correctness/completeness analysis methods

- Formal Specification
- Automated Documentation Production

D. Investigate other supporting technologies

- Methods to better communicate/affirm requirements
- Methodologies for very large systems
- Considerations for distributed systems

SS-121
Establish Controls on Software Environments

The Army must develop a viable approach to the management of software engineering environments it uses or permits to be used for MCCR development and support. Initiative and productivity of developing contractors needs to be encouraged, yet the Army must ensure that systems are supportable. In-house software support environments need to be standardized to achieve economies of scale, improve resource efficiencies, allow more rapid transition to a support posture, and improve productivity and quality.

RECOMMENDATIONS:

A. Establish requirements and standards for developers

- Effective support for Ada language
- Minimum toolset capabilities tailored to system type
- Contractor unique tools meet data representation/interface standards
- Use of specified run-time environments on target computers

B. Develop requirements for extensible Army support environment

- Army should buy instead of build tools wherever possible
- Use non-proprietary standards to form framework of system
  - -- POSIX based
  - -- Ada language standard
  - -- Design, intermediate representations
- Complete suite of life cycle tools
  - -- Prototyping and design tools
  - -- Product and process metrics
  - -- Support for host/target analysis and debugging
- Selected standard tools across all environment instantiations
  - -- Problem reporting, configuration management, ...
  - -- Contractors either use same tools or convert data upon delivery
- Insert essential development tools
  - -- Acquire with limited data rights
  - -- Encourage developing contractors to use best technology

C. Constrain target machines to meet Army needs, reduce cost

- Standard battlefield hardware
- Commercially derived family of run-time operating systems

SS–122
Manage the Introduction of Ada into the Army

Ada introduction plans and activities need to be strengthened if the efficiencies and economies of Ada are to be achieved. The use of Ada will reduce the number of tools required in the Army's support environment, improve productivity, and increase quality of software produced. No Army strategy for the control of Ada and its introduction has been evident. An effective and purposeful approach is needed.

RECOMMENDATIONS:

A. Fund an Army supplement to the OSD ATIP program

- Technology Demonstration
- Lessons learned database
- Increased technical confidence

B. Evaluate efficiency and utility of commercially available tools

- Compilers
- Program Design Language support
- Syntax Based Editors
- Dynamic debugging tools
- Code Review and Assessment

C. Develop complete Ada training program within Army

- Ada for Project Managers
- Ada contracting concerns
- Design and development using Ada

D. Evaluate success of Ada Insertion

- Cost impacts on programs
- Analysis of product quality implications

SS-123
Establish mechanism for Reverse Engineering

Standards for computer resources including software lanaguges, hardware design, documentation, and configuration control have evolved since its first application to weapon systems. In addition, many existing systems were developed in a schedule driven, resource constrained environment. Because of these factors, the Army must recognize the need to use reverse engineering to understand system design from existing software and documentation.

RECOMMENDATIONS:

A. Address need for reverse engineering in planning support

- Determine if cost effective to support in – house
- Plan on decreasing sustainment levels as system matures
- Identify specific tasks for each system

B. Establish criteria for level of reverse engineering

- Anticipated extensive software modifications
- Extraordinarily high number of software deficiencies
- Planned intensive hardware improvements
- Planned replacement date
- Evolution to common hardware/software systems

C. Investigate use of evolving technology to assist

- Transition ongoing high risk areas to Ada
- Test automatic derivation of design from existing code
- Recover knowledge – base as design is redeveloped

SS-131
Develop a Strategy for Technology Insertion

The Army must improve its software state-of-the-practice to meet the needs of the large and complex mission critical computer systems of the future. These improvements must be promulgated within the legal, fiscal, and contractual constraints of the government and reduce the risk to system development accruing from the use of unproven technologies.

RECOMMENDATIONS:

A. Identify specific risk management funds for software

B. Fund parallel developments when introducing new technology

C. Provide contract award for successful technology application to

- Improve productivity
- Improve quality

D. Establish transition points and mechanisms

- Software Technology Center as technology advocate
- Consider technology insertion in computer resource planning

E. Develop techniques for Software Process Improvement

- Software Acquisition in a Hardware NDI Environment
- Management of Firmware as if it were Software
- Assignment of Data Rights
- Structuring of Realistic Software Incentives
- Streamlining of Documentation Requirements
- Clear Communication of User Requirements
  - Formal, Executable Language for Prototypes
  - Language Understood by User, Buyer, and Builder

SS–132
Conduct Integrated Software Planning

CRMPs do not serve their intended function as currently prepared because they do not address critical issues and are not integrated into the system acquisition planning process. Planning for software must be addressed from the total life cycle viewpoint, with proper attention being given not only to initial development, but also to the critical aspects of software maintenance and improvement.

RECOMMENDATIONS:

A. Streamline Computer Resource Management Plan

- Limit size of document; remove extraneous and redundant data
- Make CRMP part of Acquisition Strategy
- Define all computer resource and funding requirements
- Define Hardware/Software Acquisition and Support Strategy

B. Computer Resource Working Group provides forum for PM

- Include LCSEC, testers, evaluators
- Provide early visibility into system strategy

C. CRMP documents conditions of eventual software support

- PM identifies resources to be programmed
- LCSEC guarantees ability to support if strategy executed

D. Approval of CRMP by PEO/AAE ratifies strategy

- MACOMs provide body of experts to advise PEO
- Feedback on effect of decision provided to AAE

SS–133
Tailor Software Acquisition Process to Systems

The Army needs to encourage the use of alternative software development models rather than the rote application of existing standards. The vast differences in the software that the Army buys as well as the limits of the "waterfall model" must be recognized and deliberate steps taken to reduce acquisition risk. Procedures are needed to: refine requirements prior to design, strengthen the design process, emphasize "software first," clarify design parameters, and improve the user/developer interface.

RECOMMENDATIONS:

A. Establish multi – axis acquisition classification scheme, including

– Degree of experience with similar systems

– Size of system

– Sensitivity of system to doctrinal change

B. Define candidate strategies for different system classifications

– Life Cycle Model

– Software Environment acquisition strategy

– Requirement stability

– Software Reuse Potential

– Contract and Support strategy

– Evaluation strategy

C. PMs classify systems and use classification to structure acquisition

D. Ensure risk areas addressed before Full Scale Development

– Prototype hardware/software design

– Trace design back to users requirements

– Base decisions on timing, storage, performance measurements

SS-134
Develop a Consistent Contracting Approach

All too often, software received late consideration in the contracting process. The time to establish specific requirements, get contractor commitments, and ensure adequate resource allocations is prior to contract award. Procedures need to be established to reward competent contractors, force an early consideration of development plans, and negotiate effectively for software consideration during development.

RECOMMENDATIONS:

A. Address software in proposal preparation instructions

- Provide software plans as part of technical approach
- Define specific government requirements in SOW
- Implement plan in proposal; do not buy as DID

B. Evaluate contractor software maturity in source selection

- Detailed evaluation of SEI process model by gov't experts
- Establish level below which contractor considered non – responsive

C. Incorporate software performance as part of MACOM database

- Evaluate Contractor's past Performance
  - – Previous Software Developments
  - – Dedication to Total Quality Management
- Include Information in Ongoing AMC Database Development "Evaluation of Contractor Past Performance in Source Selection"
- Identify "Blue Ribbon" Software Contractors
  - – Consider in Source Selection
  - – Recognize Outstanding Performers

SS–211
Organize Army to Manage Acquisition Process

Make realignment on Army staff to provide effective Army Acquisition Executive control over the acquisition of Army systems, especially those which rely on mission critical computer resources. Clear management control is needed to: improve management practices, unify the DA staff into an efficient structure, and develop a credible advocate for computer resources to Congress and the national leadership.

RECOMMENDATIONS:

A. Eliminate bicameral MCCR management at HQDA

- Establish full – time Acquisition Exectuive for Army
- Consolidate Acquisition Management in Single Organization
- Focus MCCR Policy in Acquisition Office
- Establish Expertise in Real – Time, Command&Control Systems

B. Correct AR 70 – 1 so it applies to information handling systems

- Define Applicability IAW Chapter 8 of AR 70 – 1
- Correct AR 25 – 1 so it Excludes MCCR

SS–212
Improve PM/PEO Computer Resource Management

Establish an effective working relationship with closer cooperation between the PM/PEO and their supporting MACOMs. The present system, which has given PEOs a perceived independence from MACOM policy and guidance, must be changed if weapon system software management is to be improved.

RECOMMENDATIONS:

A. Dual hat functional commanders as Program Executive Officers

B. Create CRWG early to identify problems in CRMP building

- Append minutes of CRWG meetings to CRMP
- Include other services on Inter – Service Systems
- Establish CRWG prior to Milestone I

C. Use CRMP as sole basis for computer resource strategy decisions

- Eliminate duplicative waiver/approval Processes

D. AAE establish process to stop systems with ill – conceived CRMP

- PM certifies no embedded Computers used if no CRMP
- Require/Review/Approve CRMP prior to each Milestone Decision

E. Hold LCSE directors responsible for raising planning deficiencies

F. Use "contracting authority" as required

- Structure RFP to acquire Software intelligently
- Influnce Source Selection process to consider Software
- Prevent awards, if necessary, if process goes awry

G. Provide experts to advise PEO; report to MACOM/AAE

SS–213
Establish Clear Organizational Responsibilities

Provide for a clear understanding of organizational responsibility at all levels within the Army. Changes are needed to: delineate the roles of the major organizational elements in the area of computer resources, implement a cost effective and cohesive organizational structure, provide clearer lines of management within the Army, and prevent duplication of effort in the various organizations.

RECOMMENDATIONS:

A. Define role of HQDA in overall management

- Establishment of policy objectives
- Advise the AAE on specific acquisition program decisions
- Advocacy for resources

B. MACOMs enforce policy and define strategy

- Maintain body of expertise to assist PM/PEO
- Identify systemic problems and provide corrective actions
- Promulgate policy based on lessons learned
- Establish procedures for consistent Application of Technology

C. Major Subordinate Commands support acquisition

- Support PM's acquisition and provide field support
- Evaluate Ability to Provide Support for Emerging Systems
- Maintain infrastructure to Support Transitioned Systems
- Execute supporting technologies program, as assigned

SS-214
Strengthen AMC's Software Management Role

The AMC organization needs to take into account the importance of mission critical computer resources to the Army. The command must manage its computer intensive systems so they are reliable, meet user requirements, and are supportable during their life cycle. In order to do so it should be resourced to manage the increasing role of computer resources in weapon systems, provide a strong advocate on the AMC staff, and provide career paths for software professionals.

RECOMMENDATIONS:

A. Establish well resourced, limited life Special Operations Center

- Execute a well – defined Charter
  - – Task Assignment
  - – Define Authority and Supporting Organization
  - – Provide Sunset Clause
- Provide for Mission to be Assumed by permanent Organization

B. Create an ADCS for MCCR

- Policy assistance and surrogate for HQDA
- Management of LCSE
- Provide expert advise/lessons learned
- Track computer resource trends and build strategy
- Resource Advocate

C. Establish senior level MCCR S&T advisor to Commander

SS-221
Provide One-Stop Support for Project Managers

The scarcity of computer hardware and software experts within the Army makes it critical that the available people are used effectively, provisions are made to nurture and develop a competent staff, and functional duplication is eliminated. The Life Cycle Software Engineering Centers should become the responsible activity to ensure that this happens. As such, the must become the single source of software support for PMs.

RECOMMENDATIONS:

A. LCSE **responsible for in-house software engineering**

- Requirements prototyping
- Computer Resource Planning
- Contractor evaluation and selection
- Independent Verification and Validation

B. LCSE **provides services to PMs, not a "body shop"**

- Focus is on Products for PM
- Center Director Responsible for Quality of Product
- LCSE Provides Environment to Develop Software Competency

C. **PM/PEO staffs limited to managers not doers**

D. **PMs/LCSE ensure software visibility during development**

- Provide Visibility into Formal Unit & Integration Tests
- Enhance Information Flow to PA&T, AMSAA, TECOM, OTEA

E. LCSE **maintenance activities under rigorous controls**

- Integrated Configuration Management program
- Internal Software Quality Program
- Subject to Process Review by Product Assurance

SS-222
Build an Army Software Technology Center

The trend to disperse the critical mass of technologists supporting software and to decrease the annual research and development budget for software technology must be reversed. The Army needs an integrated, effective approach to software technology which will provide a critical mass for software tools and technology, serve as a vehicle for technology insertion, and insure responsiveness to Army wide MCCR needs.

RECOMMENDATIONS:

A. Reaffirm decision to have STC at Ft. Monmouth

B. Establish critical mass of people and funding for 5 years

- Initial R&D Budget of $15M/year
- Assemble Staff of 100

C. Establish resource source, concentrate other activities

D. Create a software technology affiliates program

E. Run Software Engineering Intern program as part of STC

F. Define specific technology insertion tasks and controls

G. Assign technology proponency to STC; advocacy at HQ

H. Develop technology program with maximum use of commercial base

SS–223
Organize to Grow Software Engineers

Recognize that software engineers have different skills and abilities than others. Army must plan to grow its own Software Engineers from within and also needs.to ensure their effective use. Provide a mechanism to provide both technical and domain maturity before putting Software Engineers into management positions.

RECOMMENDATIONS:

A. Use new GS–0854 series; do not permit grandfathering

- Establish Tough Qualification Standards for 854s
- Require Engineering and Computer Education/Experience

B. Provide four distinct levels of performance

- Intern: Formal training program for technical development
- Apprentice: Develop domain experience at LCSEC
- Journeyman: Spread talent to HQ, PM, PA&T, ...
- Senior: Key management decision positions

C. Establish Opportunities for Senior Software Engineers

- Require PM System Engineers to have Software Competence
- Create MCCR Software Positions at HQDA & MACOM HQs
- Develop "Software Chief Engineer" Positions

D. Use co–op program to Identify Outstanding Candidates

- Encourage feeding of co – op Employees into Inter Program
- Early bonding with Organizational Leadership

SS-224
Eliminate Confusion in Training Device Support

Because transition of life cycle support for training devices and systems has been difficult to achieve, AMC must ensure that an organizational structure is in place to provide the life cycle software engineering support. A solution needs to take into account the problems of resourcing, support to system specific devices, interoperability, and the inherent difficulty in building a software support capability.

RECOMMENDATIONS:

A. Assign life cycle software engineering responsibility as follows:

- TRADOC
  - – Systems used at TRADOC
  - – Courseware which is separable from system software
- AMC
  - – System specific devices assigned to same LCSEC as system
  - – Generic systems to CECOM center at Ft. Leavenworth

B. Guide process by following rules

- Designate LCSEC for each specific system using above guidelines
- LCSEC integrally involved in development process
- Training device developer programs resources

C. Test use of Total Contractor Software Support as alternative

- Perform cost benefit analysis of concept
- Specify documentation as priced option to reduce risk

SS-225
Provide Virtual Colocation with TRADOC Centers

Where AMC and TRADOC centers are colocated, the communication between the two is generally excellent. Where the centers are physically separated, communication suffers. Communication in a wide variety of areas must be improved. Areas of importance include: problem identification and tracking, requirements understanding, configuration management, and test participation.

RECOMMENDATIONS:

A. Use electronic means to provide virtual colocation

- Electronic Mail
- Video Conferencing
- Electronic blackboards

SS – 231
Develop Pilot Software Awareness Program

The Army needs to publicize: (1) real and near real – time software's pivotal role in fulfilling Airland Battle Doctrine, (2) software engineering enabling role in developing and maintaining efficient, effective, and eccnomical combat software. Awareness of software's force multiplication aspects will support resource allocations at the highest level of government.

RECOMMENDATIONS:

A. Develop Airland Battle Software Story

- Why turn operational battlefield to software
- How software enables fulfillment of Army future needs
- How software is providing combat force multiplier
- Why tactical software forms ever increasing part of Army's budget
- Initiatives Army is taking to control software cost and quality

B. Prepare an Airland Battle Softwara awareness brief

- Pilot interactive video software awareness program
- introductory video tape
- Briefing slides/viewgraphs and script

C. Brief key Defense leaders on software role/Initiatives

- Congressional members and staffers
- OSD, ARSTAF, MACOM, and MSC leaders
- General officers throughout Army

D. Solicit and record feedback information

- Clarity and impact of briefing's message
- Capability to visualize, implant, and sustain importance of software

SS-232
Develop Operational Software Literacy Program

Army needs to develop an Airland Battle Software Awareness/Literacy program for congressional, OMB, OSD, Joint, and Army leadership which will: (1) elevate their consciousness .evel with respect to software's pivotal role in winning the Airland battle in the 1990 and beyond timeframe, (2) address software awareness/literacy within the Officer and NCO Corps, and (3) support the harnessing of GI creative potential in using software as a force multiplier.

RECOMMENDATIONS:

A. Build Software Literacy program based on Pilot Awareness Effort

- Expand Army's Airland Battle Software Story
- Develop interactive Software Literacy Program
- Develop interactive Software Engineering Programs
- Update Video Tape
- Update Briefing materials

B. Execute Literacy Program which includes

- Train - up Active Army, ANG, and USAR
- Spark creativity of Officer and NCO Corps with regard to software
- Use GI insight to influence Software System Engineers

C. Get to General Oficers to show impact of software

- All General Officers Army wide to become literate
- Briefings should be presented by software knowledgeable GO

SS-233
Find Army Software Advocates

Computer technology budget has decreased by order of magnitude in last five years. An advocate is needed at both the MACOM and ARSTAF levels. Additionally, proponency for Life Cycle Software Engineering appears confused with weapons system support having no effective proponent.

RECOMMENDATIONS:

A. Find fighters for software technology at AMC & DA

B. Correct failure of AMC proponent to support MCCR

- Separately identify and track MCCR software support
- Provide for MCCR representation at HQDA Budget Panels
    - – Recognize that IM proponent supports MIS/ADP
    - – Tradeoff between MIS/ADP and MCCR at Appropriate Levels

SS-311
Establish clear Acquisition Policy for Software

The Army should provide a clear, unambigous implementation of DODD 5000.29 for Mission Critical Defense Systems. Chapter 8 of AR 70 – 1 establishes the basis for such a policy, but it needs to be implemented and remaining ambiguities with the AR 25 – series regulations needs to be removed. Realistic policies and controls applicable to PEOs and PMs need to be implemented.

RECOMMENDATIONS:

A. Establish clear and concise definition/process for MCCR

B. Create integrated policy stream under AR 70 – 1 for MCCR

C. Integrate computer resource issues into PM/PEO/AAE process

- Use CRWG to help PM build strategy
- LCSEC responsible for early identification of problems
- MACOMs provide experts to help PEO review/evaluate plans

D. Require all approvals and waivers in single document

E. MACOMs maintain database on computer resource requirements

F. Provide implementation in AR 70 – series regulation

- Revise associated regulations simultaneously
- Put detailed technical considerations in DA Pamphlet

G. Require consideration of life cycle tailoring

H. Provide guidance for evolving new paradigms/environment strategy

SS–312
Clarify Funding Policy for Software Support

Need to obtain clear – cut and unambiguous guidance on LCSE funding policy that will provide the most efficient management of LCSE functions. Recent funding policy changes have streamlined the process, but several residual issues must still be addressed.

RECOMMENDATIONS:

A. Determine where to report manpower needs in BPRR/MAMP

- Conversion from OMA P2 to OMA P7M scheduled for FY 90
- Need to avoid separation of dollars and spaces

B. Consider augmentation of OMA core funding in MDEP MS2B

- Reimburse OMA with RDTE & OPA based on ratio of core tasks
- Collect RDTE & OPA funds by increasing task overhead

C. Limit use of MCM for software improvements

- When associated with improvements to hardware, or
- When specific dollar threshold is exceeded
- Otherwise use OMA P7M process as defined in current policy

SS–313
Provide for Management of Software Change

Individual software changes to systems need to be identified, costed, prioritized, and approved through a disciplined change approval process. Although costs for systems will be estimated based on the best available models, the OMA P7M funds which are identified need to be expended to get the best possible value to the Army. A joint prioritization must serve as the basis for allocation of funds and identification of deferred software maintenance and improvement tasks.

RECOMMENDATIONS:

A. Classify software support tasks into two simple categories
- Maintenance: correction of deficiencies
- Improvements: new capabilities added

B. Define a minimum level of sustainment for each system
- Define minimum levels to maintain Supportability for each system
  - – Determine cost of maintaining Support Environment
  - – Consider need to maintain expertise in Unique Languages
  - – Assess quality of Documentation/Software Structure
  - – Level should decrease as function of learning curve
- Consider sustainment needs in prioritizing work across systems
- Determine time to cease support on case – by – case basis
  - – Logical point to freeze configurations
  - – Statistical Confidence that critical errors removed

C. Conduct an annual joint AMC/ISC/TRADOC prioritization
- Identify and cost out each proposed change
- Merge maintenance and improvements into one master list
- Prioritize all proposals and rank 1 – to – N
- Fund appropriate improvements through MCM process
- Allocate funds in priority order to remaining changes
- Re – allocate if necessary to maintain sustainment of selected systems

D. Complete review early so that funds can be reprogrammed
- Identify impacts of funding shortfalls
- Terminate support cleanly as required
- Don't plan on reestablishing support after it has been interrupted

SS–314
Establish Internal Controls and Feedback

Internal controls within MACOMs need to be used to minimize the risk of having software materiel weaknesses. In general, existing controls have failed to provide feedback and corrective actions. Actions have primarily been driven by outside audits, studies, and reports. Each MACOM should establish a management and control process to identify and correct systemic weaknesses regarding the development and support of mission critical software.

RECOMMENDATIONS:

A. Establish a formal process to identify/investigate problems

- Identify system problems
- Create mechanism for problem feedback
- Develop lessons learned
- Implement corrective actions

B. Create database to track software issues/problems/solutions

- Identify specific software related issues
- Classify issues into problem areas
- Identify solutions to problems
- Remove items from database after solution effectiveness shown

C. Assign responsibility and demand accountability

- Establishing corrective action system
- Execution of specific recommendations

SS–315
Develop a Computer Resource Data Base

Today's major problems with software development are not basic technology problems, but failures in management. A major re – examination and change in attitudes and practices concerning software acquisition is needed. A key part of that change in attitude must be a more comprehensive view and assessment of the computer resources used in MCCR systems.

RECOMMENDATIONS:

A. Develop a database for system computer resource information

- Establish compatible databases at each MSC
- Define criteria used to determine how MCCR are entered
- Identify key resource information
  - – Host and target hardware
  - – Languages used
  - – Design methodologies/tools used
  - – Software development/support environment characteristics
  - – Funding information to support budget formulation
  - – Identification/cost of system change proposals
- Use as basis for command management analysis/reports

B. Establish capability to feed MSC database via DDN

- MSC maintain data from Computer Resource Plans
- Roll – up and summary data available for MACOM use
- Provide for tracking of Systems and resources
- Use to support long range planning

SS-316
Enhance Interaction between Activities

Periodic and ongoing activities should be used to improve communication and foster interchange of information between Army and other DoD activities with an interest in mission critical software. The Life Cycle Software Engineering Steering Committee should be revived to foster cooperation between Army activities. Support to the JPCG – CRM should be expanded to best utilize the cooperation between the services on policy, technology, planning, and software support matters.

RECOMMENDATIONS:

A. Re-establish quarterly meetings of Army LCSEC Commanders

- AMC, TRADOC, ISC, HSC, COE
- Resurrect old Steering Committee Charter
- Provide for:
  - – General session for information sharing
  - – Separate meetings for MACOM issue resolution
  - – Domain expert working session for specific problem areas

B. Provide regular General Officer meetings with Steering Committee

- Active participation by Army proponents
- Provide forum for problem resolution
- Formal report by Steering Committee
- Focus on policy/funding issue discussion

C. Expand support for JLC Software panel

- Use to gain leverage off other services activities
- With DARPA control of STARS, consider restart of Technology Panel
- Establish common PDSS policies and procedures across services

SS–321
Integrate Software Quality into Process

American quality organizations are typically considered "second class" operations. By focusing on engineering the quality into the design rather than the "assurance" aspects, the Army needs to force quality into a position of preeminence. We need to ensure the crcdibility of our quality organizations by using fairly senior people with solid software credentials.

RECOMMENDATIONS:

A. Fully integrate quality into software development process

- Require Developer to Implement Total Quality Management
- Provide metrics as part of software environment

B. Hold LCSEC responsible for managing software development

- Support PM in development of acquisition strategy
- Provide product oriented management assistance to PM
- Conduct IV&V with in – house experts
- Ensure early identification of problems; information sharing

C. Require LCSEC to establish internal quality controls

- Establish quality standards
- Conduct design reviews and code audits

D. Hold PA&T responsible for process oversight

- Adequacy of contract provisions
- Process Evaluation
  - – Hardware/Software Development Process
  - – Integration of Hardware and Software
  - – Component, system, and qualification testing
  - – LCSEC process evaluation
- Identify systemic problem areas
- Materiel Release / Software Version Release
- Fielded System Reviews

SS-322
Improve Software Configuration Management

Configuration control of software and management of those configurations has been based on existing hardware regulations as implemented by the various subordinate commands. No standard configuration accounting systems or even software numbering systems have been selected. Standardization activities need to be pursued.

RECOMMENDATIONS:

A. Work toward a MACOM standard configuration management tool

- Select commercial tool
- Integrate into standard software support environment
- Provide standard implementation procedures

B. Institute a standard Computer Program Identification Number

- Supplement NSN which only identifies media
- Maintain compatibility with other services
- Assign LCSEC responsibility for CPIN assignment/management

C. Implement standard tiered interoperability control board

- Enforce Configuration Control over interfaces
- Within BFA and between BFAs
- Develop capability to model and test interfaces

D. Facilitate Software Reuse

- Establish repository for reusable parts
- Issure standards/criteria for included software
- Provide strong Configuration Control

SS-323
Implement Effective Interoperability Control

Army needs to enforce a top-down approach to develop, plan, and refine baselines to support a "system of systems" approach to interoperability. Concepts which are now stated at a high level must be refined to define, model, evaluate, and control system to system interfaces. Interoperability evaluations cannot be deferred until operational tests. A hardware basis is needed for component integration below the level of the command and control nodes.

RECOMMENDATIONS:

A. Create Army Interoperability Executable Model

- Use to evolve detailed specifications from high level requirements
- Support variety of levels of specification
- Simulate message loading/reconfiguration

B. Accelerate funding of Army Interoperability Network (AIN)

- Provide distributed C3I test suite
  - -- Simulators/Stimulators
  - -- High speed network
  - -- Links to Joint Test Beds/Testers/Contractors
- Support variety of test/evaluation functions
  - -- Development and acceptance testing
  - -- Regression and Version Certification Testing
  - -- Software readiness for Operational Test

C. Build Government interoperability knowledge base

- Collect information posessed by IV&V contractors
- Use as basis for further requirement development

D. Investigate component integration/standardization

- Address hardware standardization below C&C nodes
- Prevent multiple development of C&C software

SS-324
Address Software as part of Materiel Release

Current regulations permit, but do not specifically require use of the Materiel Release process for software. The resulting confusion needs to be resolved with a clear statement that the release process be used to release all block improvements to software.

RECOMMENDATIONS:

A. Revise AR 700-142 and AMCR 700-34 to address software

- Software to be released as block improvements
- Interoperability statement required for all software
- All software changes need to follow release procedures
- Software only releases eliminate hardware specific statements

B. Evaluate and recommend procedures for special/evolving needs

- Emergency releases
- Evolutionary life cycle model

SS-325
Develop Responsive Distribution Process

AMC tactical computer systems increased from 85 systems in 1980 to 232 systems in 1989, and will continue to grow. The standard logistics supply system is not adequate for supporting software change distribution, especially when major modifications to interoperating command and control systems must be accomplished. Current method of sending teams from the LSSEC will be impractical as the number of systems continues to grow. Alternative methods need to be investigated now.

RECOMMENDATIONS:

A. Conduct experiment with forward replication/distribution

- Establish AMC in – theater assistance center
- Electronically transmit software upgrades and documentation
- Use desktop publishing capability to prepare documentation
- Replicate software and print documentation at forward site
- Install and train from forward site

B. Develop Army go – to – war strategy for software upgrades

C. Require consideration of externally programmable memory

- Could reduce the configuration burden
- Simplify upgrade process
- Supports different software versions in different theaters

D. Develop regulation addressing Software Distribution

SS-326
Provide Software Maturity Management

Systems must be managed so we avoid a "final exam mentality." The Army does not now have, but needs to use an approach for tracking the maturity of software in systems. Deficiancies must be identified and corrective actions taken before the system reaches its formal testing phase. It is critical that the focus on system testing be lessened. The Army must ensure that component tests are properly structured and the information from them is used to identify and remove deficiencies.

RECOMMENDATIONS:

A. Develop, evaluate, and then use software metrics

- Process measures
- Product measures
  - observable behavior (e.g. time to failure)
  - design and code attributes

B. Require use of approved monitoring tools

- Emphasize "engineering" not "assurance" aspects
- Use care in applying hardware type indicators
- Select proven techniques, eg.
  - Defect density
  - Test sufficiency
  - Defect cause and type distributions

C. Use system approach to show intermediate results

- Prototype evaluation by users
- Stress testing of system components
- Early integration testing with interoperable systems
- Allow "free play" testing prior to formal test
- Don't allow schedule driven premature initiation of formal testing

D. Reach consensus for on-going evaluations

- Agree on system evaluation criteria up-front
- Encourage LCSEC, TECOM, AMSAA, OTEA participation
- Develop consensus of deficiencies
  - Engineering assessment of failure cause
  - Corrective actions

SS-411
Enforce Standard Software Cost Model Use

A variant of Boehm's COCOMO software cost estimating model has been developed for Army use in Life Cycle Software Cost forecasting. The model, called SECOMO, was validated but different versions are starting to appear. A standard, approved version should be maintained. In addition, further refinements to the model need to be addressed and a methodology for forecasting development, in addition to support costs, needs to be developed.

RECOMMENDATIONS:

A. Mandate use of one authorized version of SECOMO model

- Revalidate MEA approved model
- Provide for uniform application across LCSECs
- Verify compliance with field audits

B. Support further refinements to the model

- Retain MEA approval and certification
- Develop modifications to address Ada cost differences
- Implement knowledge – based front end
- Provide templates to address LCSEC unique aspects

C. Determine areas where further improvements are necessary

- Collect actual cost data
- Assess actual against predicted requirements

D. Conduct research to develop model for use in development

- Data Requirements
- Model Development

SS–412
Improve Interface into PPBS for Software

Establish a capability to capture total LCSE requirements and latest funding guidance from multiple commands and appropriations. Isolate and track LCSE costs through the PPBS process.

RECOMMENDATIONS:

A. Design a process to capture LCSE requirements/guidance

- Consistent with Computer Resource Management Plans
- Reflect output from approved cost forecasting model
- Capture core and system specific requirements
  - – OMA direct funding
  - – RDTE/OPA for improvements under MCM process
  - – OPA for hardware environment improvements
  - – MCA projects for LCSE construction/upgrade
  - – Spaces and manpower authorizations

B. Provide timely feedback from PPBS decisions

- Identify resources to specific system needs
- Provide basis for reclama/defense

SS–413
Identify and Capture Actual Software Costs

In spite of the ever increasing cost of software to the Army, It is not possible to identify and track those costs. Actions need to be taken collect software costs both during development and during the support phase of the life cycle. It must be recognized that collecting hardware and software costs together does not provide sufficient visibility into the development process.

RECOMMENDATIONS:

A. Develop common data definitions across services

- Establish software cost data collection criteria
- Use prescribed Work Breakdown System for software
- Tri – service basis for data collection provides maximum leverage

B. Require contractors to isolate and report software costs

C. Establish standard procedures to report in – house software cost

D. Develop policies and instructions concerning cost identification

- Use other service policies as models
- Maintain historical records in Computer Resource Database

SS-421
Provide Efficient Front End Loading

The Army evolved the concept of Post Depolyment Software Support into Life Cycle Software Engineering approximately five years ago. This action provided additional consideration of software engineering at the front end of development rather than waiting until it was time for support. With more resources required to support the increasing number of transitioned systems, it is time to refocus resource allocation to emphasize early, high leverage actions.

RECOMMENDATIONS:

A. Define specific front end tasks to be done in - house

- Construction of rapid prototypes
- Contractor maturity measurement
- Hands - on review of design and code
- Internal IV&V execution

B. Determine methods to Resource Front End Tasks

C. Establish Army sponsored FFRDC for Acquisition Assistance

- Provide System Engineering Expertise
- Focus on Command & Control Systems
- Expertise in Real - Time, Embedded Computer Systems

SS–422
Consider Alternative Support Options

The concept of in – house Army software support envisioned over ten years ago was never executed because of resource constraints. Typically, each LCSE uses a support contractor to perform maintenance and improvements on the systems it manages. Sometimes government facilities are used, but other times they are not. There is a need to consider alternative support concepts with the purpose of minimizing cost and freeing up government people to focus on emerging systems.

RECOMMENDATIONS:

A. Pick selected systems to test alternative concepts

- Total contractor lifetime software logistics support
- Use of a Reliability Improvement Warranty concept
- Delivery of program generators not code; maintain at high level
- Contract Award Fees tied Directly to Field Software performance

B. Assess cost and risk of promising alternative concepts

- Have provision to acquire documentation/tools if necessary
- Conduct scientifically planned experiments

C. Establish guidelines to determine optimum concepts

- Government – Contractor Mix
- When not to implement Organic Support Capability

SS–423
Conduct Contracting Out Study

Army still does a significant amount of in – house development and support of ADP/MIS systems at Central Design Activities. These systems are much more similar to commercially available systems than those embedded in weapon systems, and the Army may be mis – allocating its people by focusing its talent on these areas while giving short shrift to its tactical systems. A complete evaluation of the feasibility of contracting out these ADP/MIS activities should be conducted.

RECOMMENDATIONS:

A. Evaluate feasibility of freeing up TDA positions for MCCR

- Formally study contracting out of ADP/MIS CDA functions
- Apply TDA surplus to MCCR oriented software needs

SS–424
Measure Efficiency of Current LCSE Centers

The decision to use a controlled number of LCSE centers is based on a study which is over ten years old. No data is available to help evaluate the effectiveness and efficiency of these centers. Productivity data should be collected and used to update PDSS concept study.

RECOMMENDATIONS:

A. Identify efficiency measures for LCSECs

- Cost per line of code changed
- Productivity measures
- Distribution of Activities

B. Institute on – going data collection effort

- Instrument Support Environments
- Provide analysis of metrics

SS – 431
Develop Software Engineering Career Program

Provide a career program with direct and tangible benefits to employees. There must be convincing evidence encouraging them to enter and stay in the field. Such a program will include the following features: strict standards to enter and progress in the program, effective career management, formal and continuing process of training and development, and good opportunities for high – level career progression.

RECOMMENDATIONS:

A. Use new GS – 0854 series for Software Engineering

B. Establish precise qualification and certification standards

C. Establish Target Jobs at various professional levels

D. Implement dual track system with equal rewards

- Technical: Research and Development; hands – on
- Management: PM/PEO, HQDA, MACOM, MSC management

E. Provide tangeable incentives at each level

- Application and academically oriented education opportunities
- Rapid promotion
- Mimimum holdover at GS – 12 levels

F. Get salary levels competitive with industry

SS-432
Improve Incentives for Military Software Experts

The Army needs to recognize the importance of Software to its war fighting capability and stop discouraging and frustrating those young officers with software talent and education. A process needs to be developed to ensure software capability is used as a criteria when assigning Program Managers to computer intensive projects.

RECOMMENDATIONS:

A. Broaden career path to General Officer

B. Provide Software Understanding for 51s

C. Eliminate 53A classification; use 25B instead

D. Provide for Functional Automators

E. Treat software intensive positions as command assignments

F. AERB identify masters degree in software for MCCR PM positions

G. Provide additional software intensive add-on to DSMC PM course

H. Accredit USMA Software Engineering Department

SS–433
Establish Career Subprogram Management

A strong career management infrastructure is necessary in order for the Army to attain maximum return on investment in Software Engineering personnel. As the job series for Computer Engineers is implemented, intensive management will be necessary to ensure that proper and effective standards are developed, only well qualified engineers are admitted to the program, each software engineer's technical and managerial maturation is planned and executed.

RECOMMENDATIONS:

A. Establish Software Engineering Subprogram manager

- Part time assignment
- Rated on measures of program's success
- Preclude slow start – up process

B. Establish network of Software Engineering Mentors

- Work with high potential co – op students
- Authorize offers – to – hire into Software Engineering Intern Program
- Work with Activity Career Program Manager
- Establish one at each MSC

C. Establish Temporary Career Management Staff

- Full time support
- Interface with Personnel
- Establish qualifications; review job standards
- Set up and administer Software Engineering Review Board

D. Write E&S ACTEDS Master Training Plan

SS-434
Provide Job Challenge for Software Engineers

Successful complex software programs use a government acquisition force 10% of the size of the contractor's software development group. Army systems seldom can muster a force this large. Need to effectively use the people we have, yet realize that they need some hands-on experience to maximize their competence.

RECOMMENDATIONS:

A. Channel Software Engineers into high leverage activities

- Concept definition, prototype development
- Software strategy, operational concepts, specifications
- source selection, program management
- Quality assurance, configuration management

B. Develop means to maintain proficiency

- Identify high-tech software intensive positions
- Rotate software engineers into high-tech positions
- LCSEC provide variety of skill building assignments
- Provide for affiliates in software research organizations

C. Identify specific skills and assess as part of IDP reviews

- Design merits of variety of software paradigms
- Portability and re-usability aspects of application code
- Evolving software methodologies
- Domain related expertise

APPENDIX C

Implementation Obstacles and Schedules

"Cheshire – Puss," said Alice, "Would you tell me, please, which way I ought to go from here?"

"That depends a good deal where you want to go to," said the cat.

"I don't care much where," said Alice.

"Then it doesn't matter which way you go," said the cat.

"... So long as I get somewhere," Alice added as an explanation.

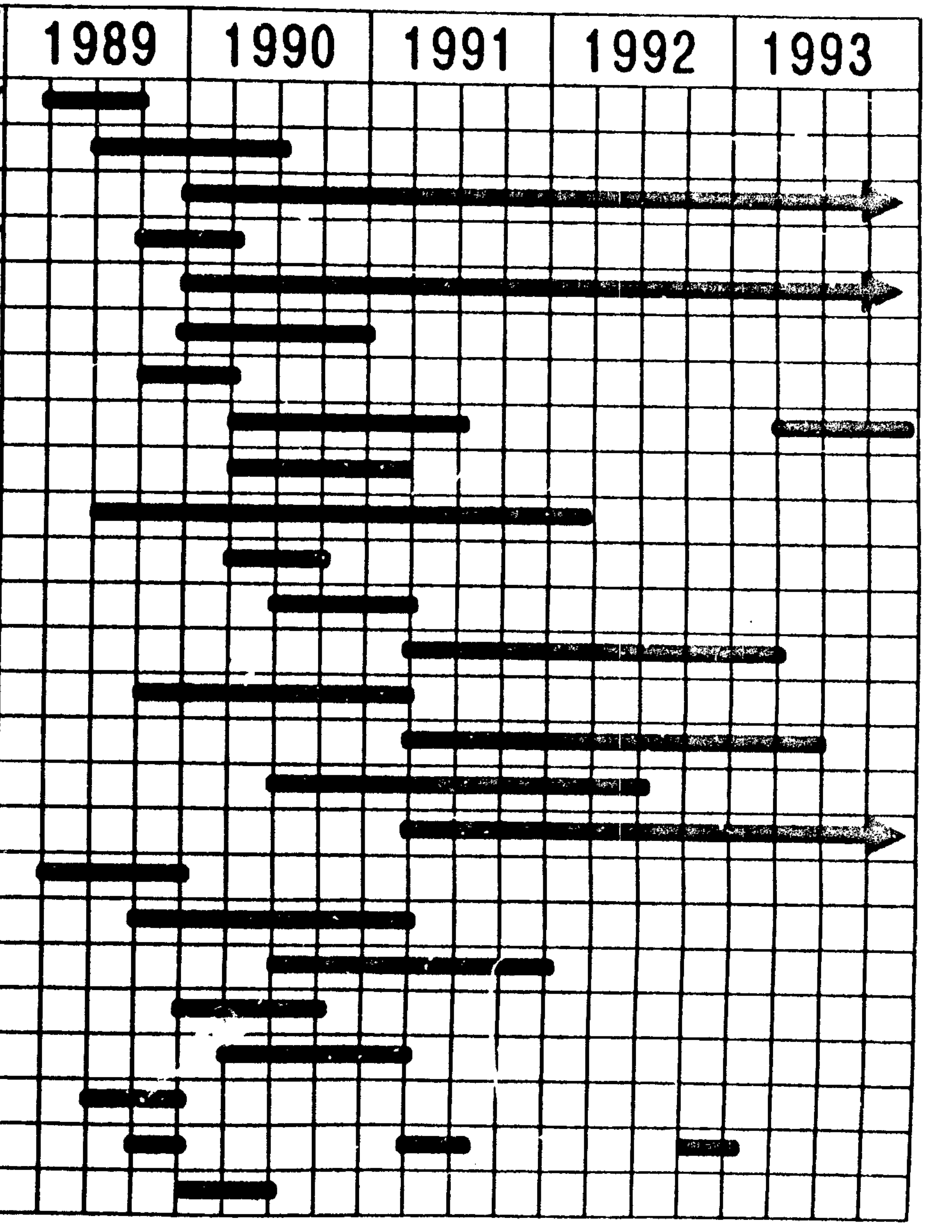
"Oh, you're sure to do that," said the cat, "if only you walk long enough."

Alice's Adventures in Wonderland

Implementation Planning Template

C-1

ID	TASK DESCRIPTION	L	R	O	I	E	P	U	T
SS-111A	Create Software Technology Plan			✓				✓	✓
SS-111B	Establish Software Testbed			✓	✓	✓	✓		
SS-111C	Evaluate Commercial Technology			✓		✓			
SS-111D	Structure R&D Program with Reuse Keystone							✓	✓
SS-111E	Research Army Specific Needs					✓		✓	
SS-111F	Structure Industry Investment Incentives					✓		✓	
SS-112A	Plan to Acquire Reusable Software							✓	✓
SS-112B	Establish Software Reuse R&D Plan			✓		✓			
SS-112C	Investigate Non-technical Reuse Issues				✓			✓	
SS-113A	Collect/use Compiler Metrics			✓	✓				
SS-113B	Establish Framework for Metric Evaluation			✓					✓
SS-113C	Calibrate Metrics with Ongoing Programs								✓
SS-113D	Develop/evaluate new Metrics					✓			✓
SS-114A	Develop model to Select SW Paradigms							✓	✓
SS-114B	Investigate Alternative SW Paradigms					✓			✓
SS-114C	Improve Specification Analysis Methods					✓			✓
SS-114D	Investigate Supporting Technologies					✓			✓
SS-121A	Establish Environment Standards				✓			✓	
SS-121B	Develop Support Environment Requirements							✓	✓
SS-121C	Establish Controls on Target Computers		✓					✓	
SS-122A	Fund Army Ada Insertion Program					✓			
SS-122B	Evaluate Commercial Tools							✓	
SS-122C	Develop Ada Training Program in Army			✓	✓				
SS-122D	Evaluate Success of Ada Insertion			✓	✓				
SS-123A	Plan for Reverse Engineering during Support				✓			✓	

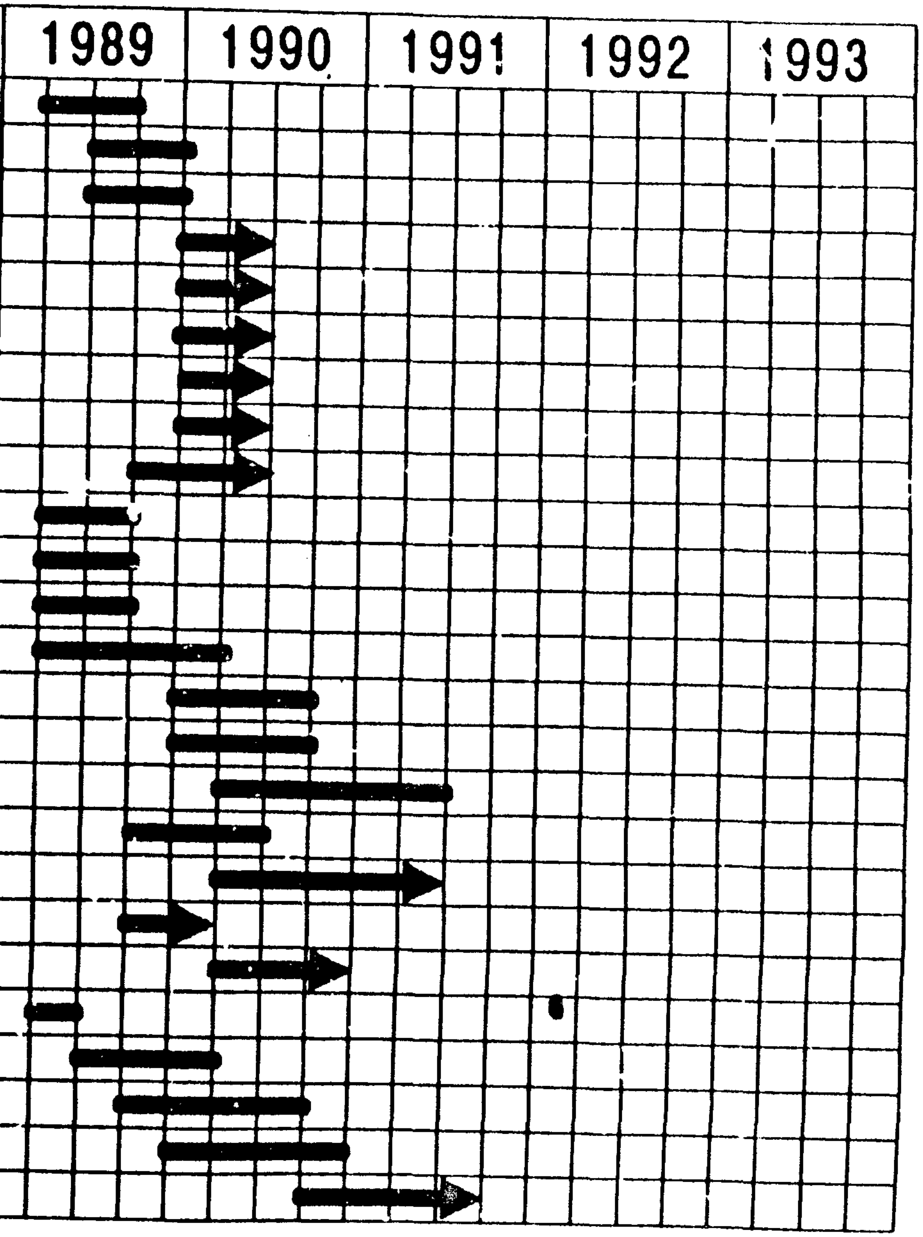


mplementation Planning Template

ID	TASK DESCRIPTION	L[illegible]	R[illegible]	O[illegible]	I[illegible]	F[illegible]	P[illegible]	U[illegible]	T[illegible]	1989				1990				1991				1992				1993			
SS-122B	Establish Reverse Engineering Criteria				✓			✓								■	■	■	■										
SS-122C	Investigate Reverse Engineering Technology								✓							■	■	■	■	■	■	■	■						
SS-131A	Identify Risk Management Funds					✓								■	■														
SS-131B	Support Appropriate Parallel Development					✓										■	■					■	■						
SS-131C	Make Technology Factor for Contract Award							✓					■	■															
SS-131D	Plan for Technology Insertion in Programs			✓		✓									■														
SS-131E	Develop Process Improvement Techniques				✓			✓							■	■	■	■	■										
SS-132A	Define Streamlined CRMP Planning Process		✓		✓						■	■																	
SS-132B	Ensure CRWG Helps PM Prepare CRMP		✓										■																
SS-132C	Use CRMP to Identify Resources/Strategy		✓										■																
SS-132D	Provide for Formal CRMP Approval/Feedback		✓										■																
SS-133A	Establish Acquisition Classification Scheme							✓						■	■														
SS-133B	Define Candidate Strategies for Classifications							✓	✓						■	■	■												
SS-133C	Provide for Tailored Acquisition Structures		✓					✓										■											
SS-133D	Require Risks Addressed before Development		✓		✓	✓											■												
SS-134A	Address Software in Proposal Preparation							✓				■	■																
SS-134B	Evaluate Contractor Software Maturity				✓			✓					■	➔															
SS-134C	Use Evaluations of Past Software Performance			✓				✓			■	■	➔																

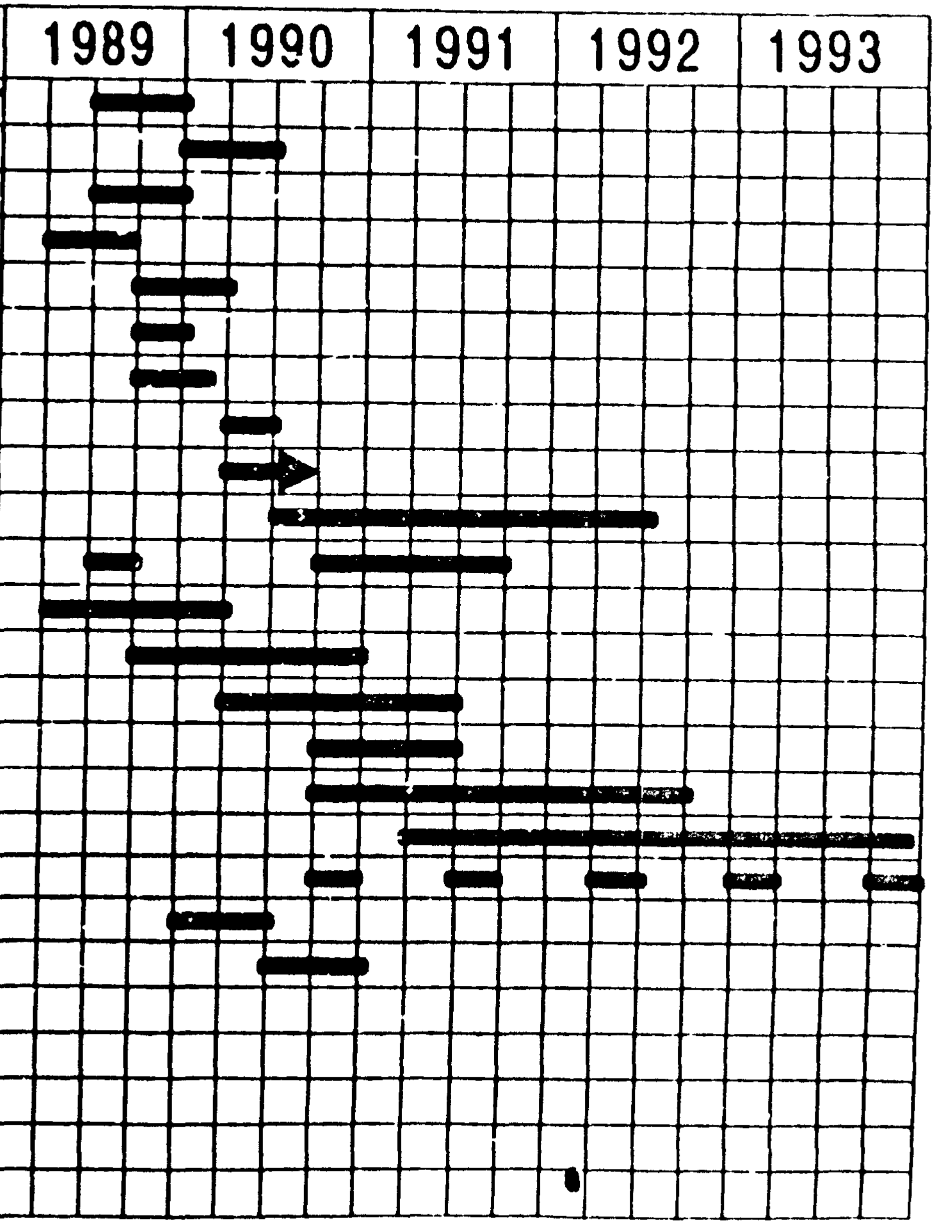
Implementation Planning Template

ID	TASK DESCRIPTION	L	R	O	I	F	P	U	T
SS-211A	Eliminate Bicameral MCCR Management		✓	✓					
SS-211B	Correct AR 70-1 Applicability		✓						
SS-212A	Empower Functional Commanders as PEOs		✓	✓					
SS-212B	Establish CRWG Early in Life Cycle		✓						
SS-212C	Use CRMP as Only Basis for Approval/Waivers		✓						
SS-212D	AAE Use CRMP as Potential Show Stopper		✓	✓					
SS-212E	Hold LCSE Director Responsible for Problem ID		✓						
SS-212F	Use 'Contracting Authority'			✓	✓				
SS-212G	Provide Experts to Advise PEOs			✓			✓		
SS-213A	Define HQDA Role in Software Management			✓					
SS-213B	Strengthen MACOM Policy/Strategy Role			✓					
SS-213C	Define MSC Support/Technology Role			✓					
SS-214A	Provide Study Implementation Workforce			✓					
SS-214B	Create ADCS for MCCR in HQ, AMC			✓					
SS-214C	Establish Senior MCCR S&T Advisor for CG			✓					
SS-221A	Focus SW Engineering Effort in LCSEs			✓					
SS-221B	Organize LCSE to Provide Service not Bodies			✓		✓		✓	
SS-221C	Staff PM/PEOs with SW Qualified Managers			✓					
SS-221D	Require PM/LCSE to Ensure Visibility into SW			✓				✓	
SS-221E	Provide Rigorous Controls for SW Maintenance			✓	✓				
SS-222A	Reaffirm Software Technology Focus in STC			✓					
SS-222B	Define Critical Mass for 5 Year Startup Period			✓		✓	✓		
SS-222C	Find and Move Resources to STC			✓		✓			
SS-222D	Create Software Technology Affiliates Program			✓					
SS-222E	Run Software Engineering Program from STC			✓			✓		



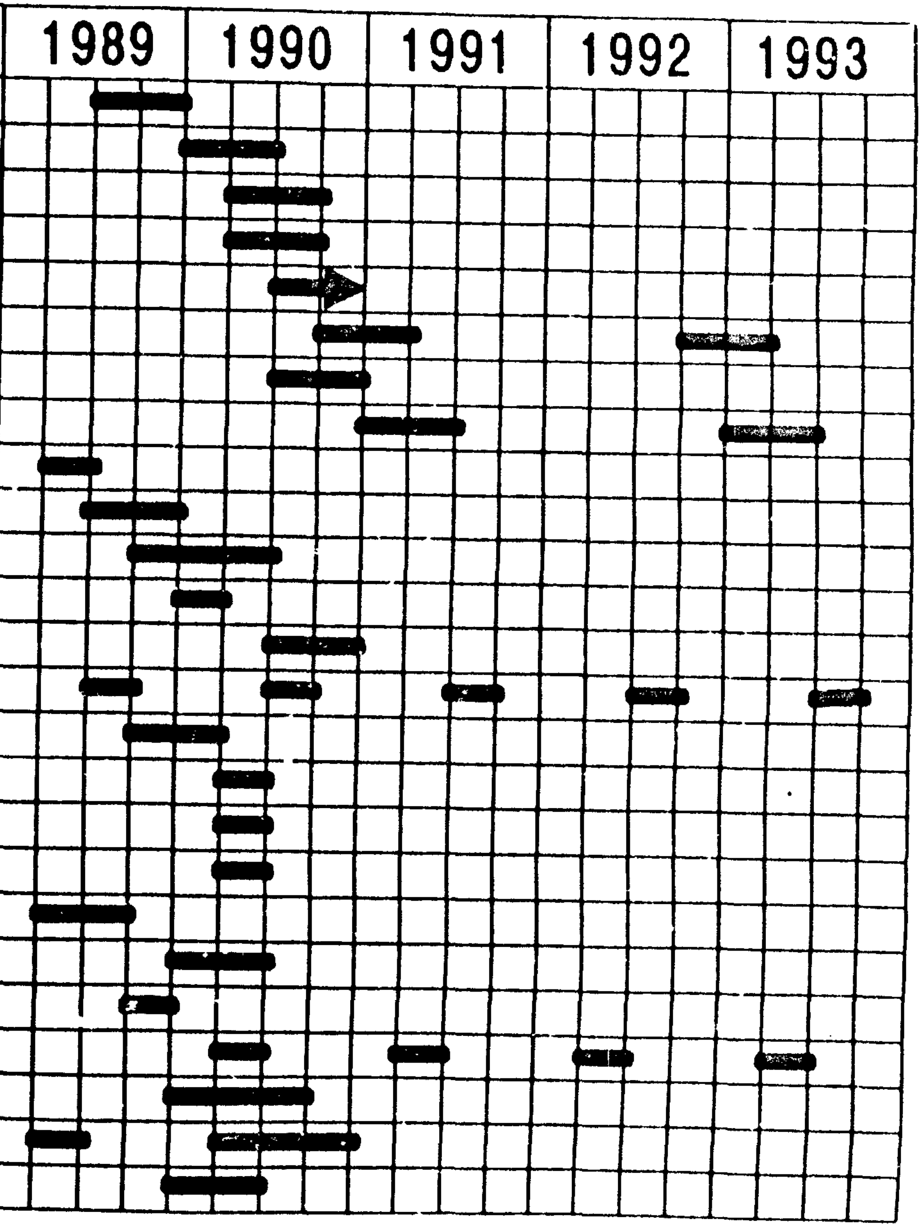
nplementation Planning Template

ID	TASK DESCRIPTION	L	R	O	I	F	P	U	T
S - 222F	Define Technology Insertion Tasks at STC			✓					✓
S - 222G	Assign Technology Proponency to STC			✓					
S - 222H	Leverage STC Program off Commercial Base			✓		✓			
S - 223A	Develop Criteria for GS - 854 Engineers			✓			✓		
S - 223B	Define Four Distinct Levels for SW Engineers			✓					
S - 223C	Establish Software Engineering Opportunities			✓					
S - 223D	Plan to use Co - op Program for SW Engineers			✓	✓		✓		
S - 224A	Assign Training Device LCSE Early/Logically		✓	✓					
S - 224B	Ensure LCSE Involvement/Funding		✓						
S - 224C	Test Total Contractor Support for PM, TRADE				✓			✓	
S - 225A	Provide for Virtual Colocation of SW Centers			✓		✓			
S - 221A	Develop SW Story/Lessons Learned			✓	✓	✓		✓	
S - 221B	Prepare Software Awareness Briefing				✓	✓			
S - 221C	Brief Key Defense Leaders on Army MCCR				✓	✓		✓	
S - 221D	Get Pilot Software Awareness Feedback				✓	✓			
S - 222A	Create Army MCCR Software Literacy Program				✓	✓		✓	
S - 222B	Execute MCCR Software Literacy Program			✓	✓	✓			
S - 222C	Provide SW Brief to Army General Officers			✓	✓	✓			
S - 223A	Identify DA/AMC MCCR Technology Advocates			✓					
S - 223B	Correct Lack of AMC MCCR SW Advocacy			✓					



Implementation Planning Template

ID	TASK DESCRIPTION	L	A	O	I	E	P	U	T
SS-311A	Establish Clear MCCR Definition		✓	✓					
SS-311B	Create Capstone MCCR Policy in AR 70-XX		✓						
SS-311C	Integrate MCCR Issues into PM/PEO Process		✓	✓					
SS-311D	Eliminate Disparate Waiver/Approval Process		✓						
SS-311E	Require Tracking of MCCR Requirements					✓		✓	
SS-311F	Integrate Implementation in AR 70-series		✓					✓	
SS-311G	Establish Requirement for Tailoring							✓	✓
SS-311H	Provide Guidance for Evolving Technologies		✓					✓	✓
SS-312A	Define BPRR/MAMP Manpower SW Process		✓	✓					
SS-312B	Consider Augmentation of OMA 7M Core				✓	✓			
SS-312C	Define Proper use of MCM Process for SW		✓	✓					
SS-313A	Classify SW Support Tasks into two Categories		✓						
SS-313B	Use Minimum Sustainment Level for Tasking					✓	✓	✓	
SS-313C	Conduct Annual Joint Prioritization		✓	✓	✓	✓			
SS-313D	Identify Specific Impacts of Funding Shortfalls			✓	✓	✓			
SS-314A	Establish Formal Problem Identification Process			✓				✓	
SS-314B	Create Database to Track Issues/Problems				✓				
SS-314C	Assign Corrective Action Responsibility			✓					
SS-315A	Develop Database for MCCR Information				✓	✓		✓	
SS-315B	Establish Capability to Network Database			✓	✓			✓	
SS-316A	Re-establish LCSE Commander's Meetings			✓	✓				
SS-316B	Provide Regular General Officer LCSE Review			✓					
SS-316C	Expand Support for JPCG-CRM in AMC			✓		✓			
SS-321A	Integrate Quality into SW Development		✓	✓					
SS-321B	Require LCSE to manage SW Development		✓	✓				✓	

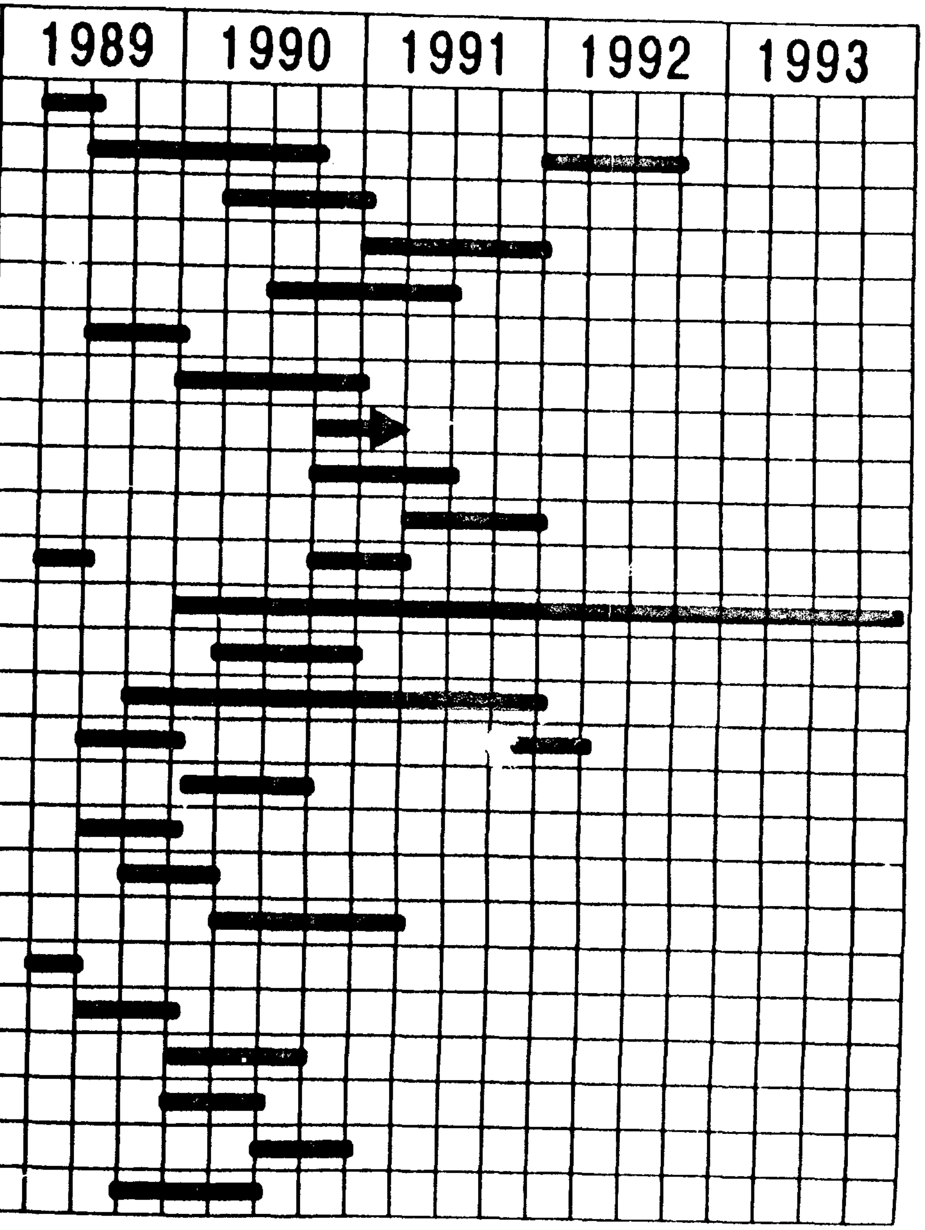


Implementation Planning

ID	TASK DESCRIPTION	L	R	O	I	F	P	U	T	1989				1990				1991				1992				1993			
SS - 321C	Require LCSE to Establish Internal SQA			✓	✓			✓								■	■	■											
SS - 321D	Hold PA&T Responsible for Oversight		✓	✓												■	■												
SS - 322A	Select Standard SW CM Tool for MACOMs			✓				✓									■	■	■										
SS - 322B	Institute Standard CPIN Program		✓	✓	✓			✓											■	■									
SS - 322C	Implement Tiered Interoperability Board			✓	✓																■	■							
SS - 322D	Facilitate Reuse Configuration Management			✓				✓	✓											■	■								
SS - 323A	Create Army Interoperability Executable Model			✓		✓		✓				■	■	■															
SS - 323B	Accelerate Funding of AIN					✓					■																		
SS - 323C	Build Government Interoperability KB					✓			✓					■	■	■	■												
SS - 323D	Investigate Hardware Integration/standards				✓			✓	✓									■	■	■	■								
SS - 324A	Revise AR 700 - 142 to Address Software		✓									■																	
SS - 324B	Develop Release Procedures for Special Cases							✓					■	■	■	■													
SS - 325A	Conduct Experiment with In - theater Distribution			✓	✓	✓		✓							■	■	■	■	■	■									
SS - 325B	Develop Army go - to - war SW Upgrade Strategy		✓	✓				✓													■	■							
SS - 325C	Require Externally Programmable Memory use							✓						■	➡														
SS - 325D	Develop Regulation for Software Distribution							✓						■	■	■													
SS - 326A	Develop, Evaluate, and Use SW Metrics					✓			✓			■	■	■	■	■	■	■	■										
SS - 326B	Implement SW Engineering Monitoring Tools				✓			✓				■	■																
SS - 326C	Require Demonstration of Intermediate Results		✓	✓				✓						■	■	■	■	■	■	■	■								
SS - 326D	Establish Process for on - going Evaluations		✓	✓									■	■	■														

Implementation Planning

ID	TASK DESCRIPTION	Law	Bdg	Org	Inr	End	Peo	Usd	Tch
SS-411A	Mandate use of One Approved SECOMO		✓		✓			✓	
SS-411B	Further Refine SW Cost Model					✓			
SS-411C	Identify Areas for Cost Model Improvement			✓				✓	
SS-411D	Conduct Research to Determine R&D Model					✓			✓
SS-412A	Define Process to ID LCSE Requirements			✓	✓			✓	
SS-412B	Provide Timely Feedback from PPBS Decisions			✓	✓				
SS-413A	Define Common SW Costing Data Elements			✓	✓				
SS-413B	Require Contractors to Isolate/Report SW Cost			✓	✓				
SS-413C	Establish Procedures for In-House SW Cost ID			✓				✓	
SS-413D	Develop Policies/Instructions for SW Cost ID				✓				
SS-421A	Define Specific Front End Tasks for In-house		✓					✓	✓
SS-421B	Apply Resources to Front-end Tasks			✓		✓			
SS-421C	Establish Army FFRDC for System Engineering			✓					
SS-422A	Test Alternative Software Support Options							✓	
SS-422B	Assess Cost/Risk of Alternative SW Support					✓			
SS-422C	Establish Guidelines for Support Decisions							✓	
SS-423A	Evaluate use of FED-COOP to Free Manpower						✓	✓	
SS-424A	Identify Efficiency Measures for LCSE							✓	
SS-424B	Institute On-going Efficiency Data Collection			✓					
SS-431A	Prohibit Grandfathering into GS-854 Series		✓						
SS-431B	Establish Software Engineering Job Standards							✓	
SS-431C	Identify Target Job Opportunities			✓				✓	
SS-431D	Implement Dual Track Career System			✓				✓	
SS-431E	Identify Specific Rewards at Each Career Level								
SS-431F	Make Salary Competitive with Industry	✓							



Implementation Planning Template

C-8

ID	TASK DESCRIPTION	Law	Reg	Org	Inr	Fnd	Peo	Upd	Tek	1989				1990				1991				1992				1993			
SS-432A	Broaden Career Path to General Officer				✓								■	■															
SS-432B	Structure Path for Software Experts as 51s			✓	✓							■	■																
SS-432C	Enhance 53A Course			✓	✓			✓				■	■																
SS-432D	Classify Software Intensive Positions as 53As				✓			✓						■	■	■													
SS-432E	Treast Software Assigments as Command				✓			✓								■													
SS-432F	AERB Identify Masters Degree for MCCR PMs			✓				✓							■	■													
SS-432G	Provide Software Add-on to DSMC PM Course							✓						■	■	■													
SS-432H	Accredit USMA Software Engineering Dept				✓										■	■	■	■											
SS-433A	Establish Software Engineering Subprogram		✓	✓							■																		
SS-433B	Establish Software Engineering Mentors			✓									■	→															
SS-433C	Provide Subprogram Staff Management			✓							■	■																	
SS-433D	Write E&S ACTEDS Plan for Software							✓			■																		
SS-434A	Channel SW Engineers to High Leverage Jobs			✓									■	■	■	■													
SS-434B	Develop Means to Maintain SW Proficiency							✓							■	■	■												
SS-434C	Measure Progress Toward Mastering SW Skills							✓								■	■	■											